

JK



Atorney's Docket No.: 07042-152001

03CO
#6

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : ERIK HENNUM
Serial No. : 09/728,073
Filed : December 4, 2000
Title : WEB-BASED INSTRUCTION

Art Unit : 2173
Examiner : Unknown

Attention: Official Draftsman
Commissioner for Patents
Washington, D.C. 20231

TRANSMITTAL OF FORMAL DRAWINGS

Please substitute the enclosed 59 sheets of formal drawings for the corresponding drawings presently in the application.

Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: September 21, 2001

Diana DiBerardino
Diana DiBerardino
Reg. No. 45,653

Fish & Richardson P.C.
601 Thirteenth Street, NW
Washington, DC 20005
Telephone: (202) 783-5070
Facsimile: (202) 783-2331

40069408.doc

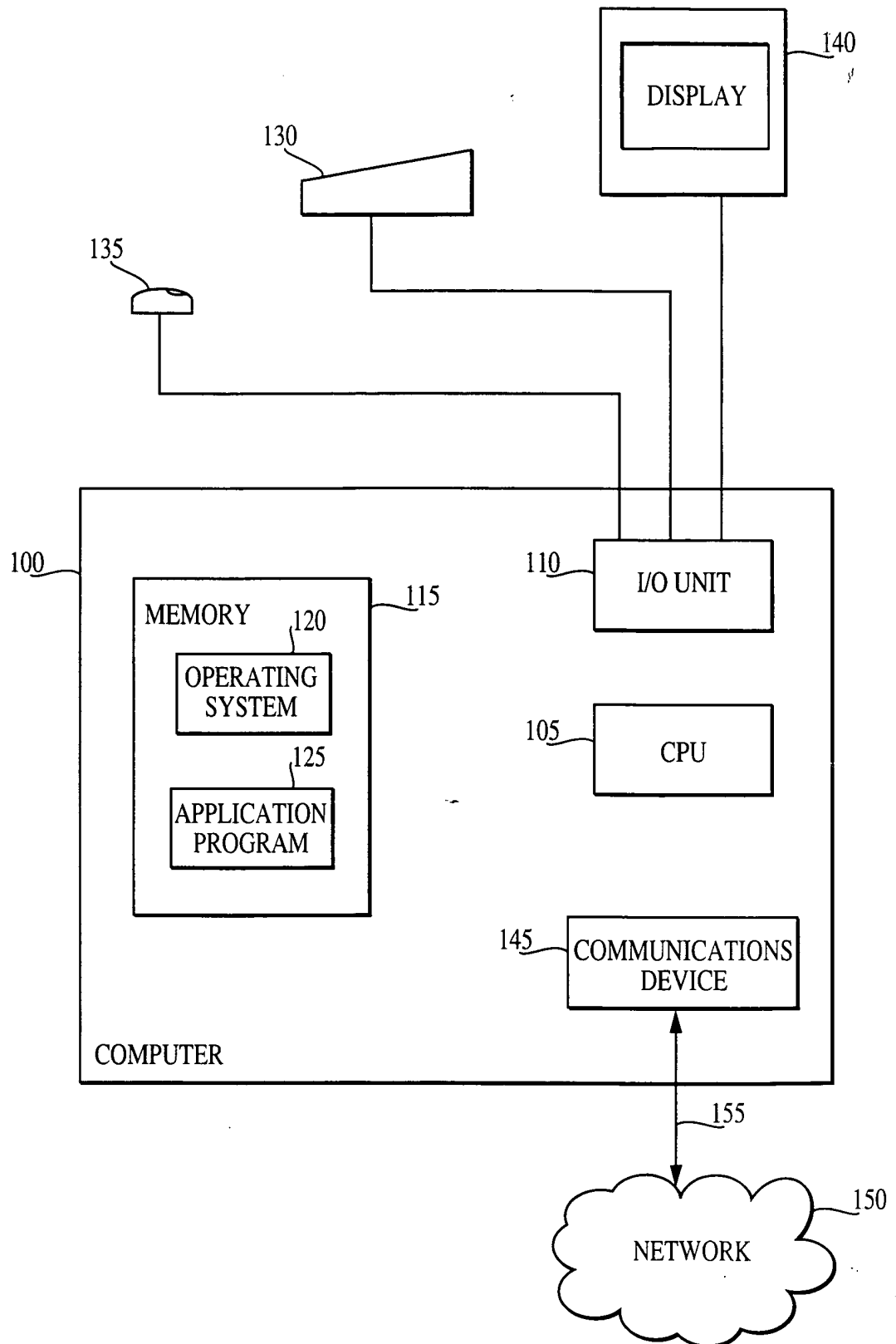


FIG. 1

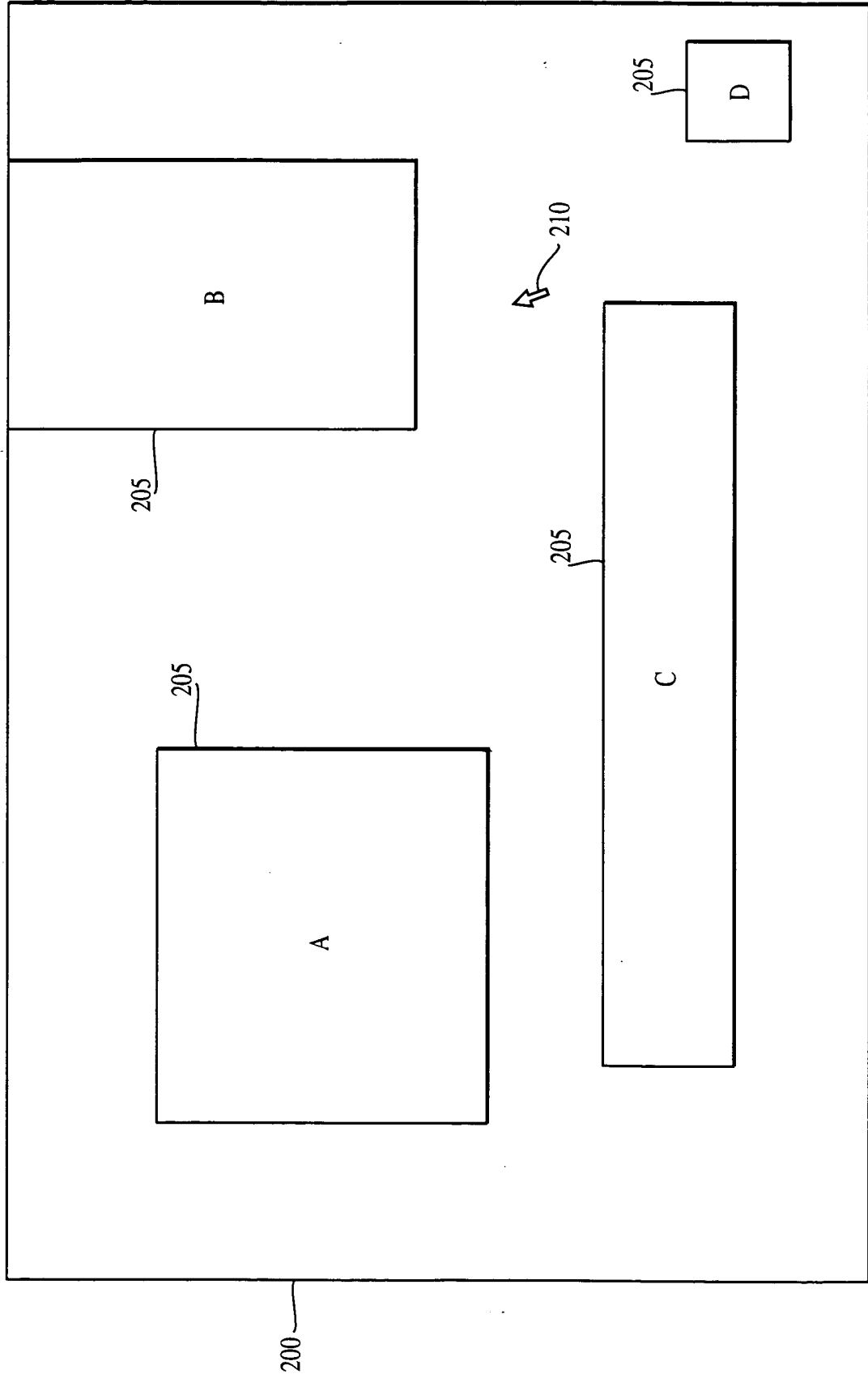


FIG. 2
PRIOR ART

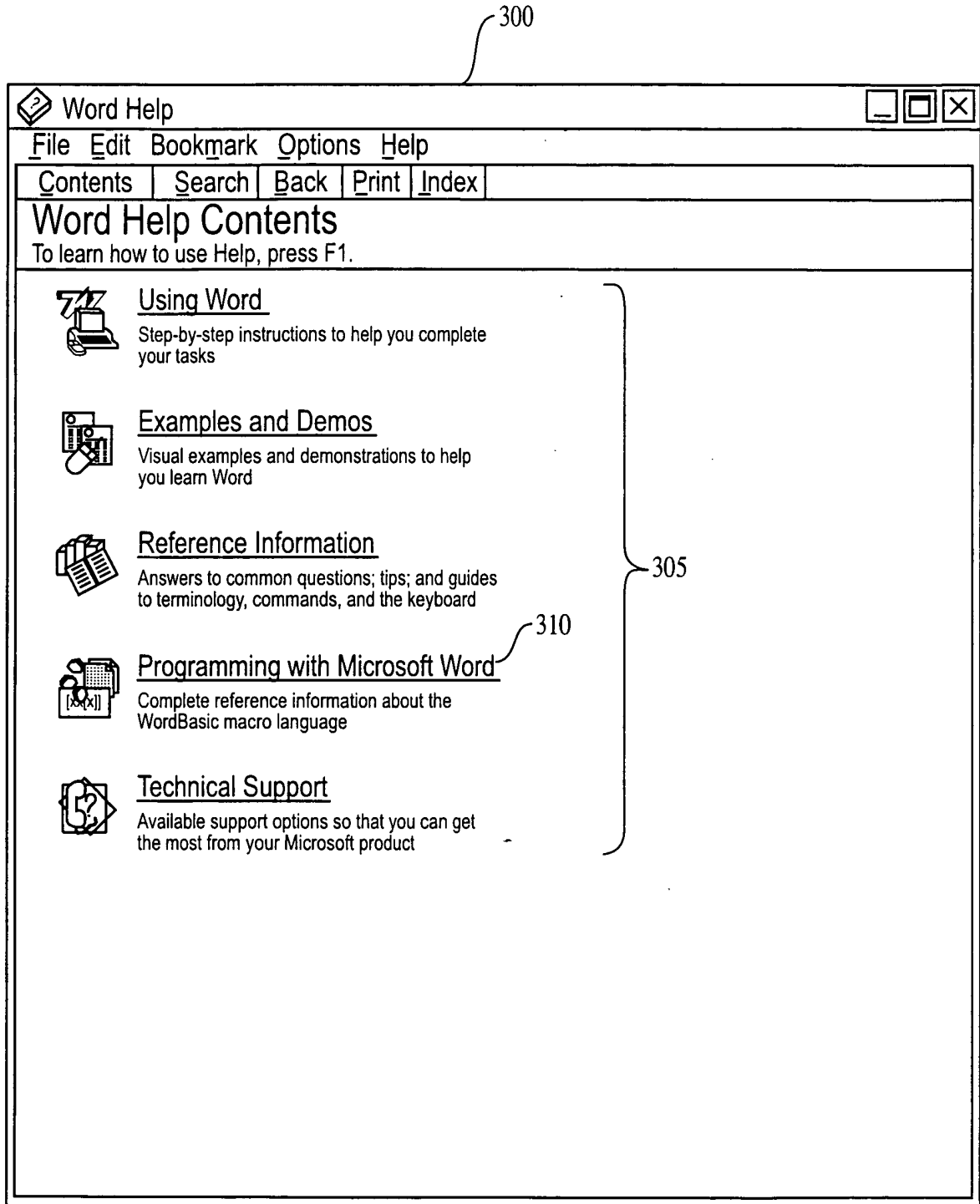


FIG. 3A
PRIOR ART

300

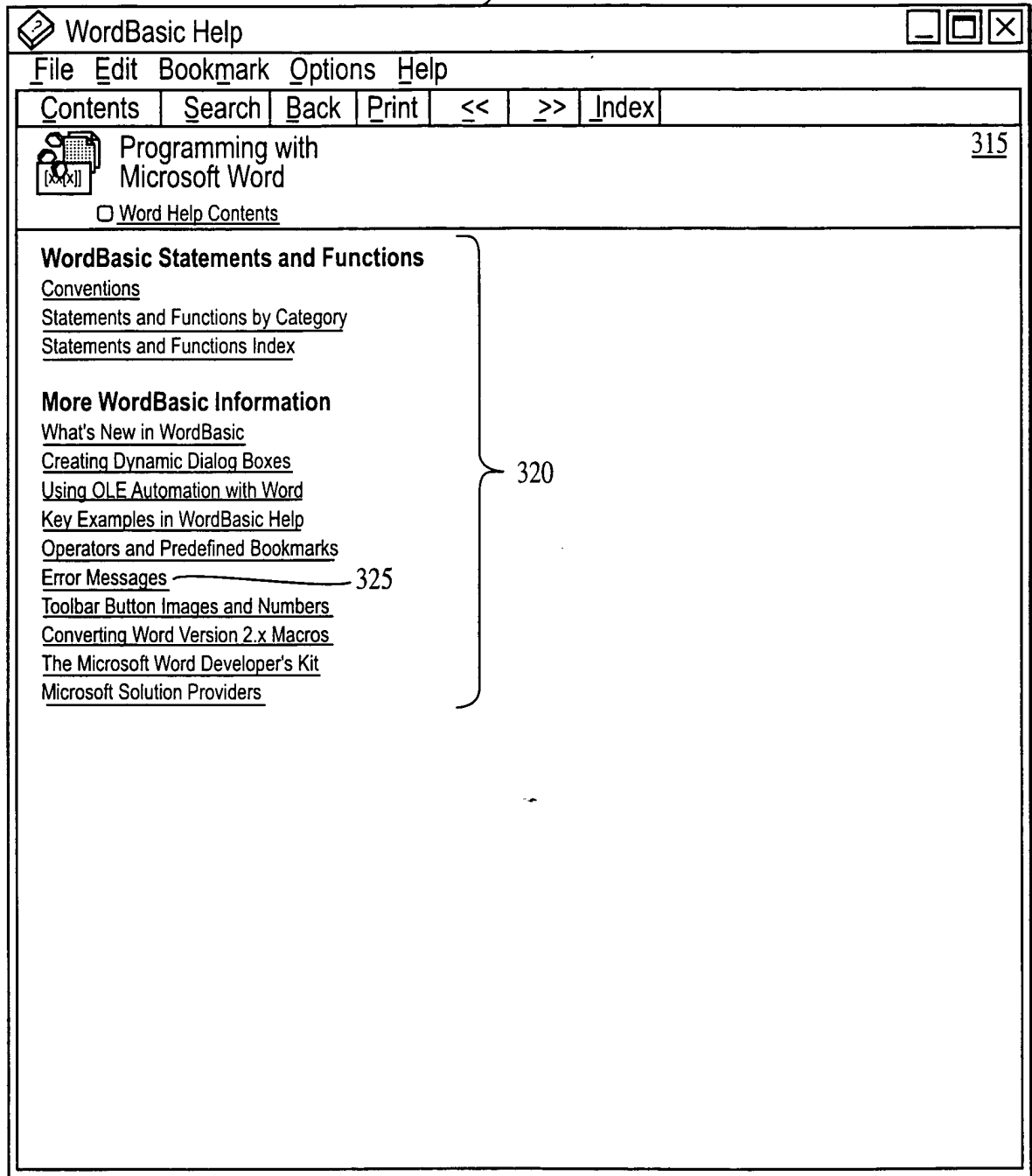


FIG. 3B
PRIOR ART

300

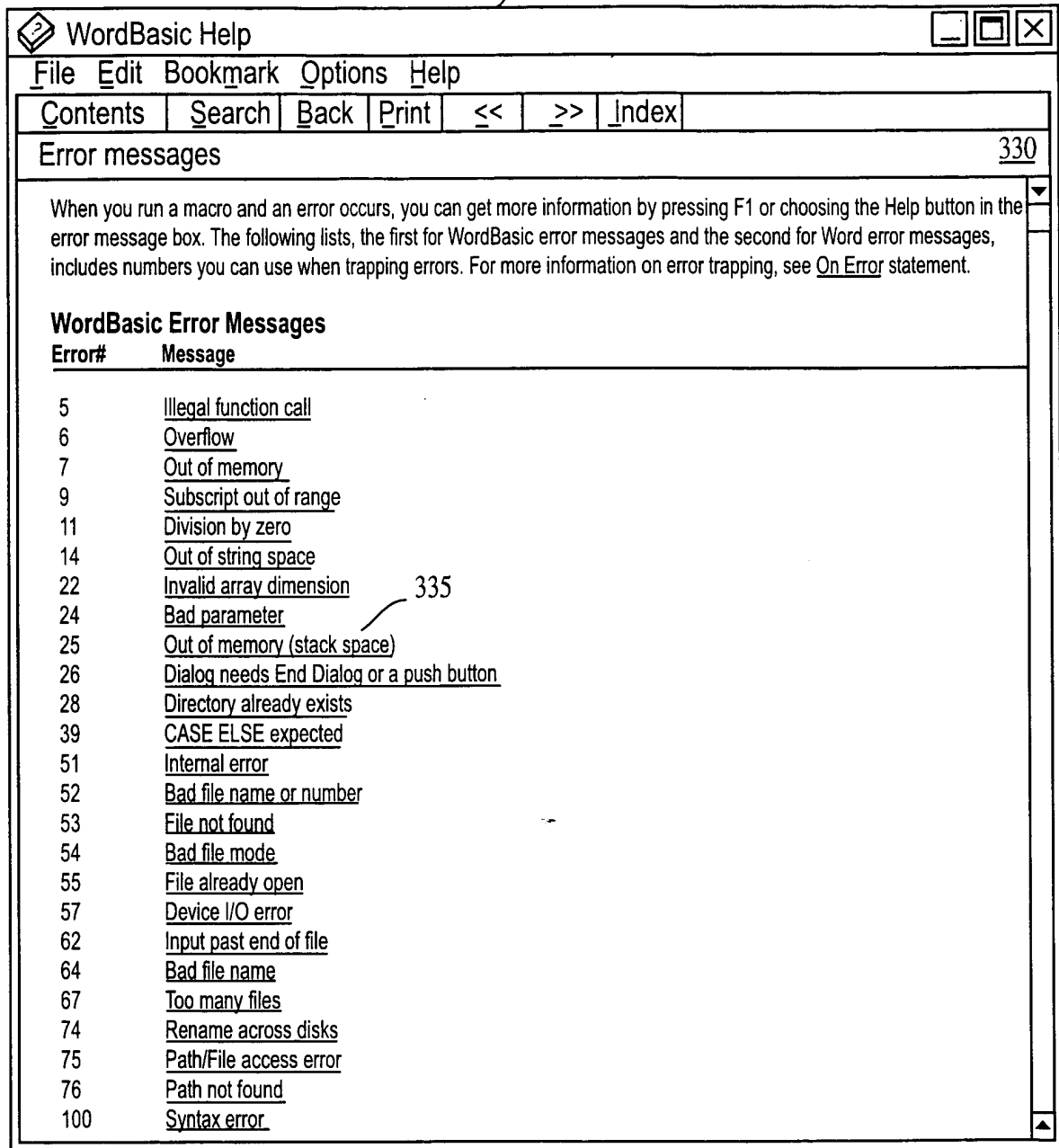


FIG. 3C
 PRIOR ART

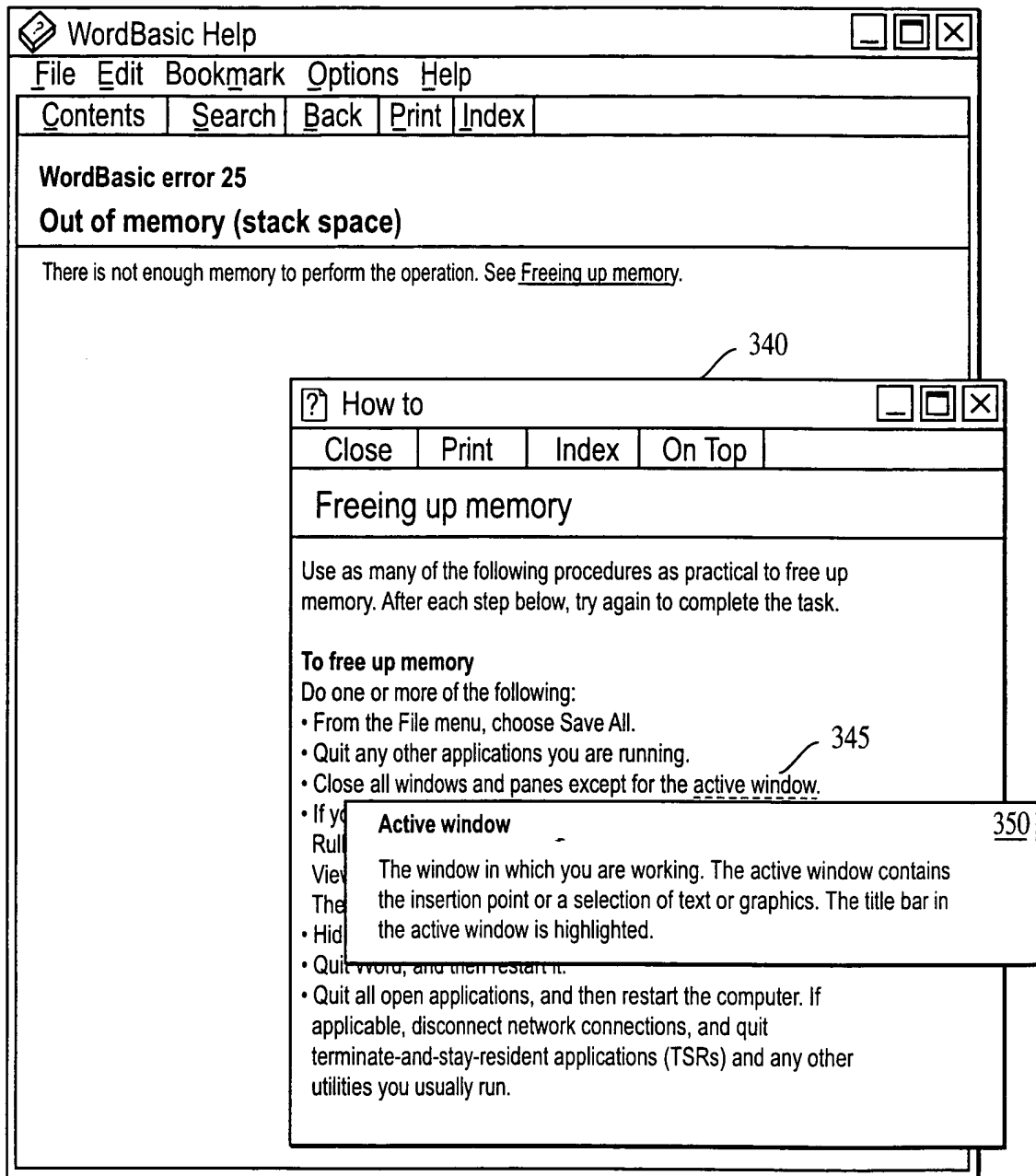


FIG. 3D
PRIOR ART

410

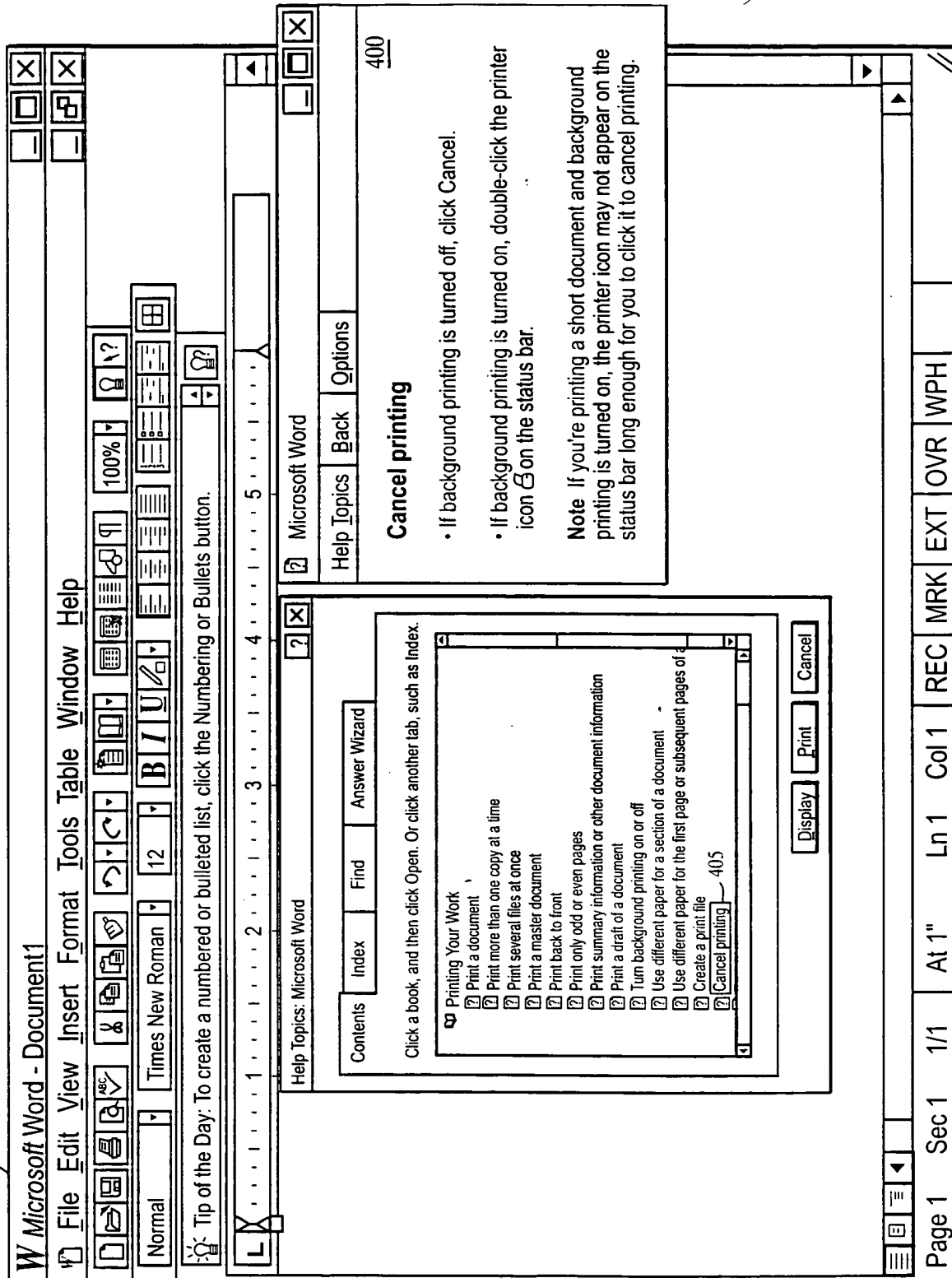


FIG. 4
PRIOR ART

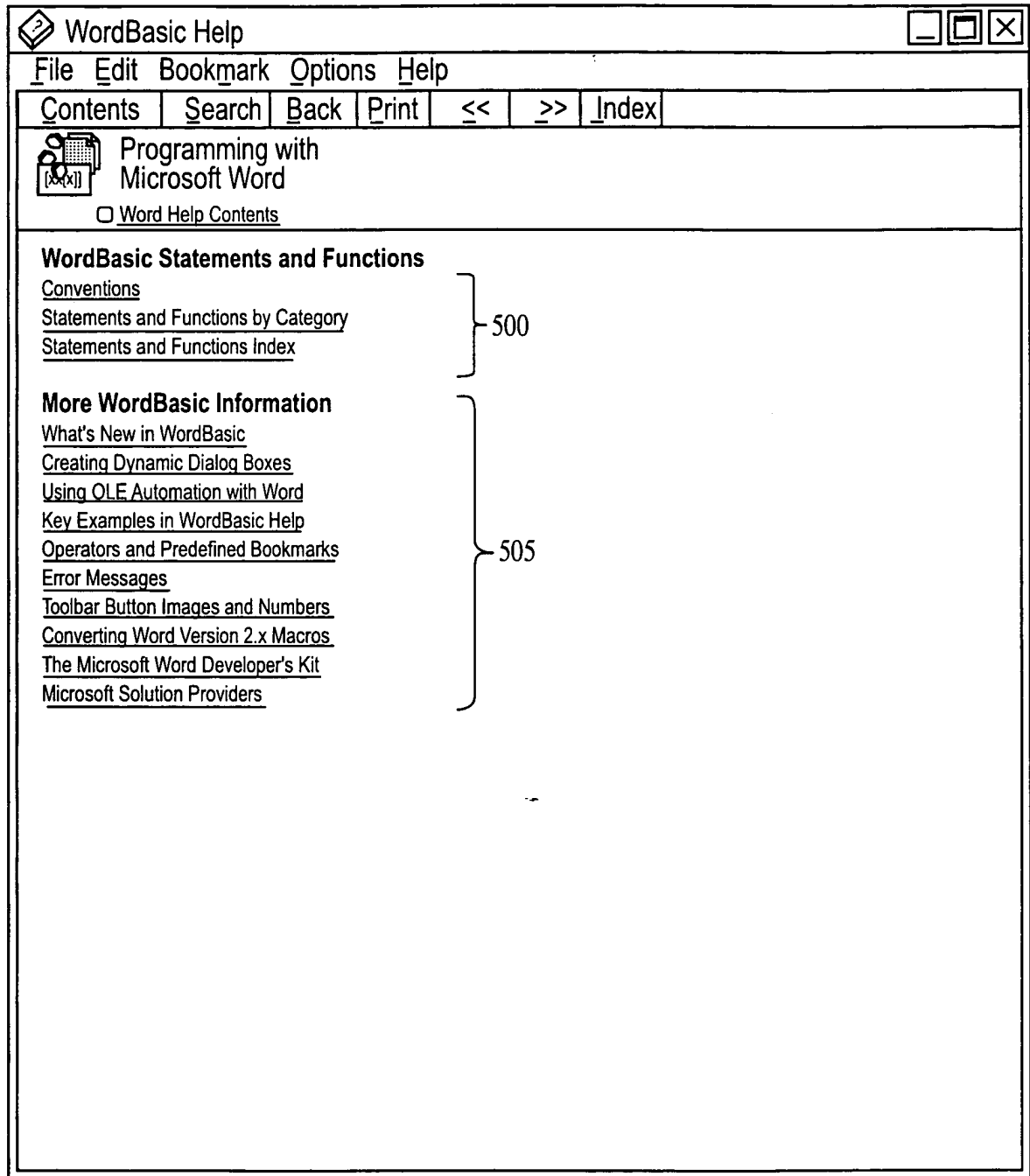


FIG. 5A
PRIOR ART

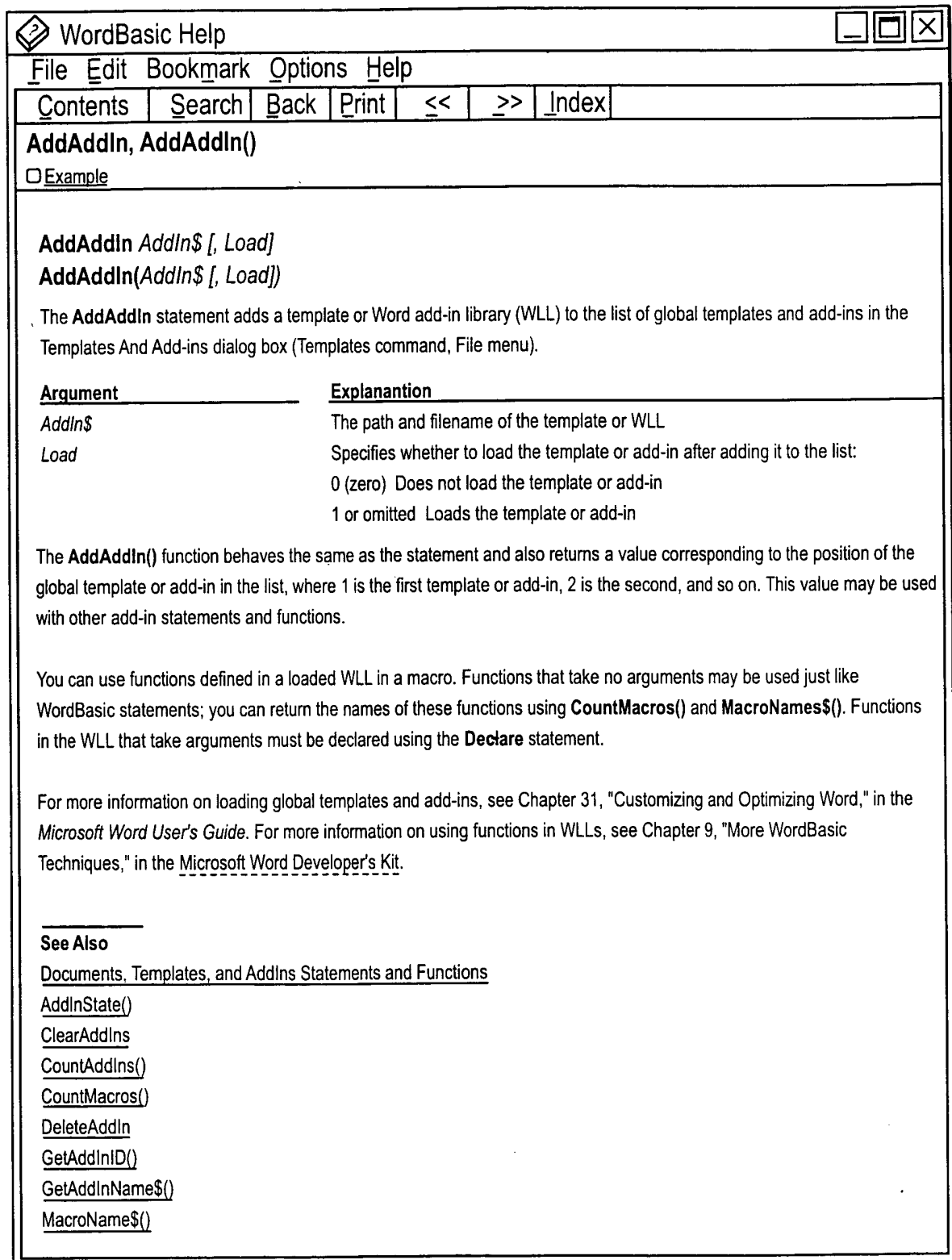


FIG. 5B
PRIOR ART

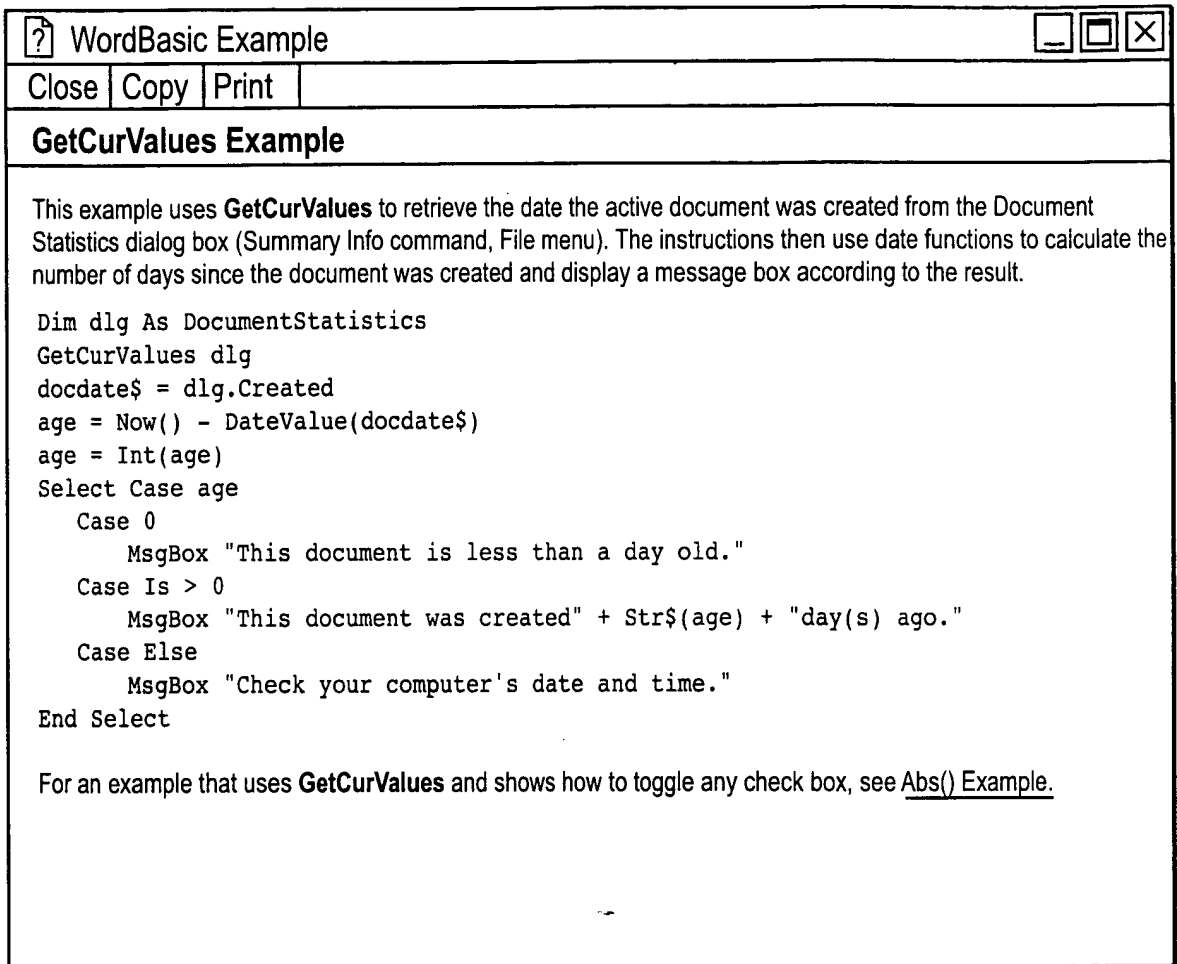


FIG. 5C
PRIOR ART



FIG. 6

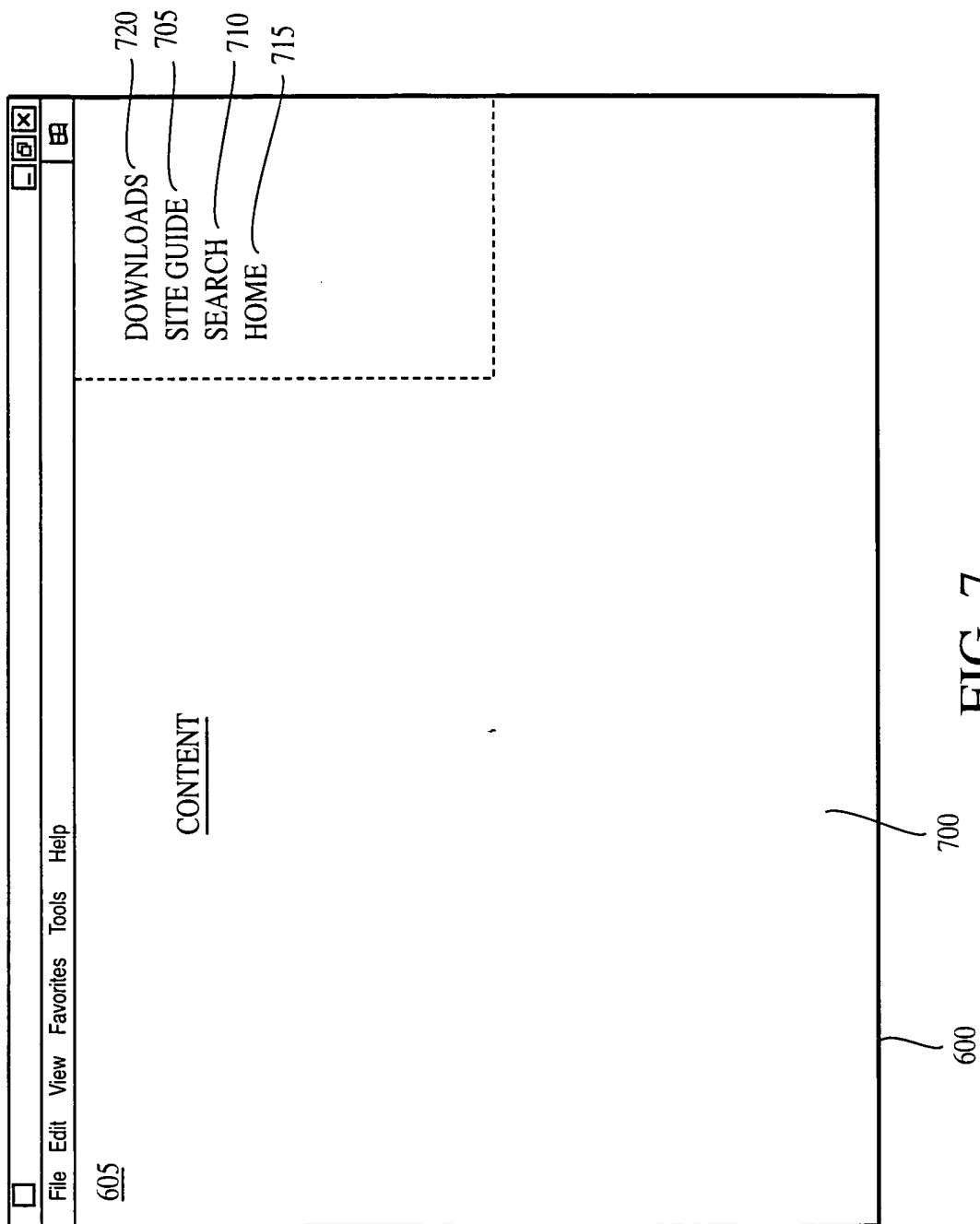


FIG. 7

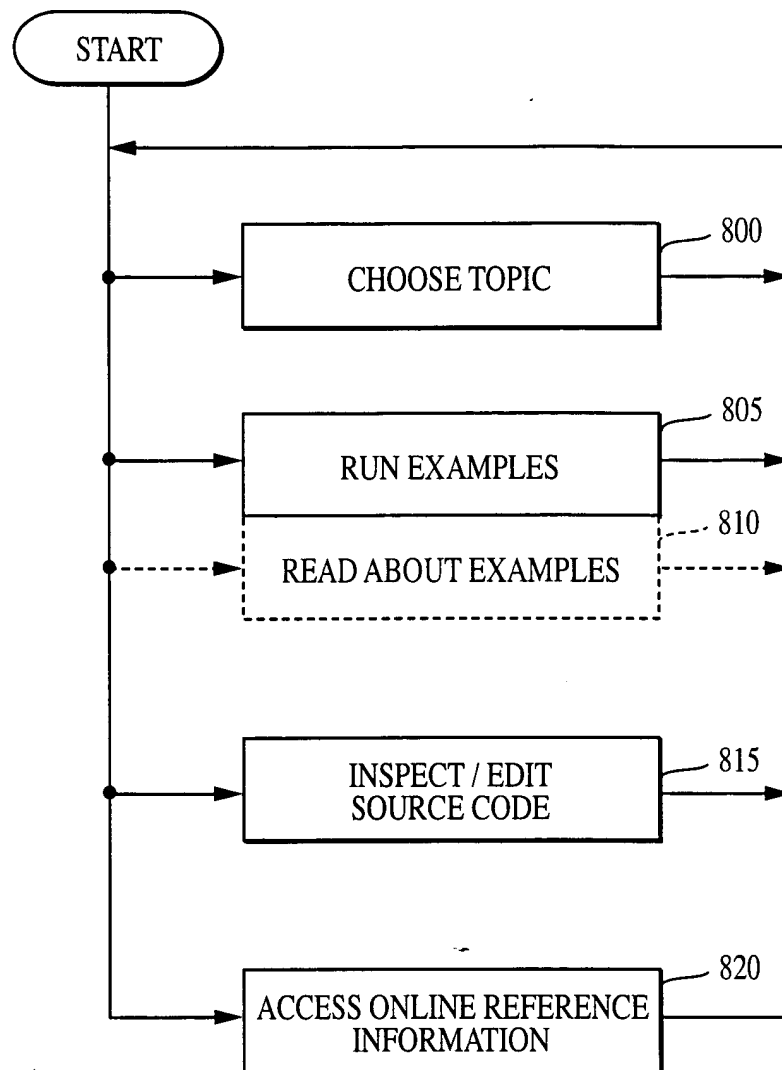


FIG. 8

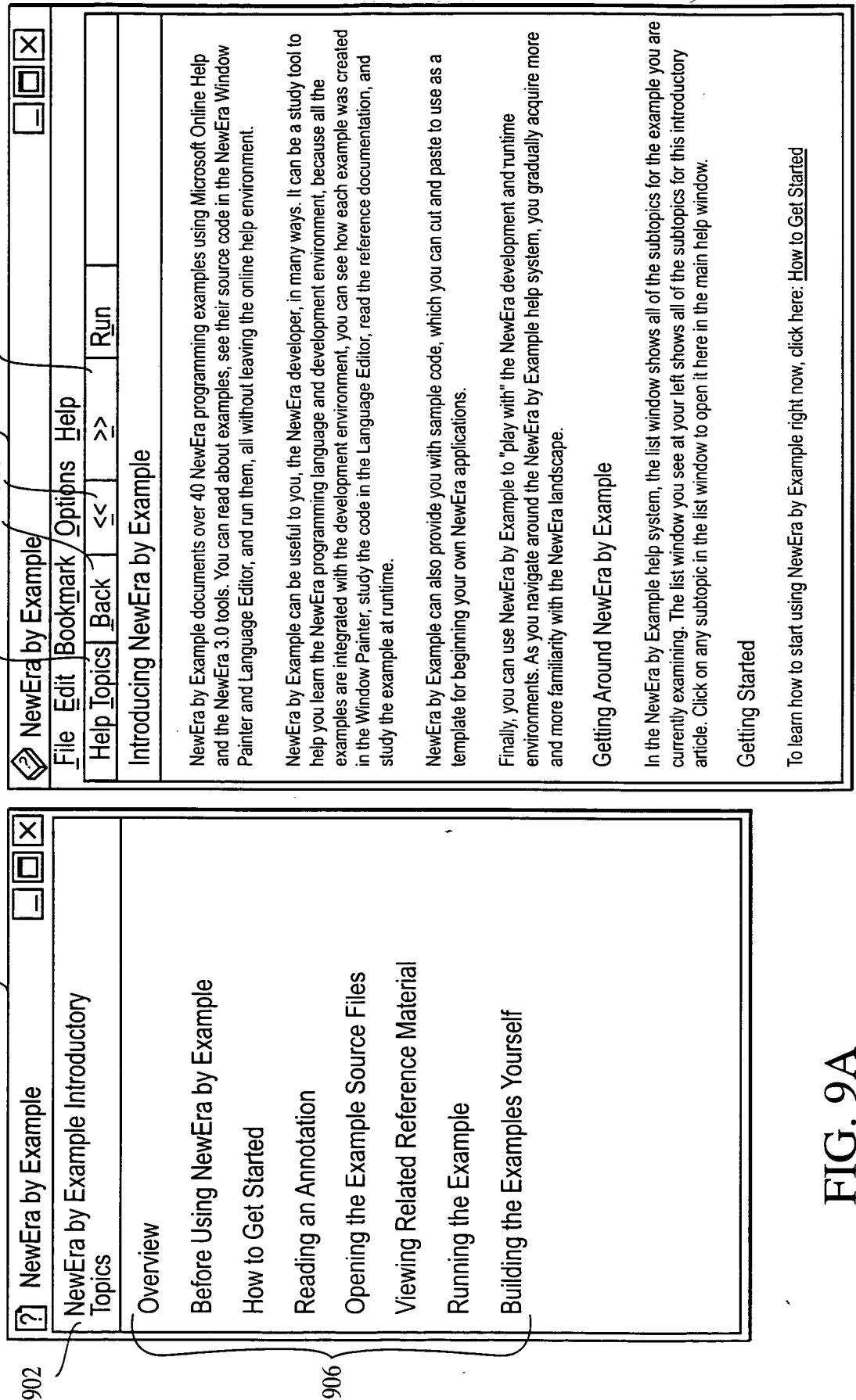


FIG. 9A

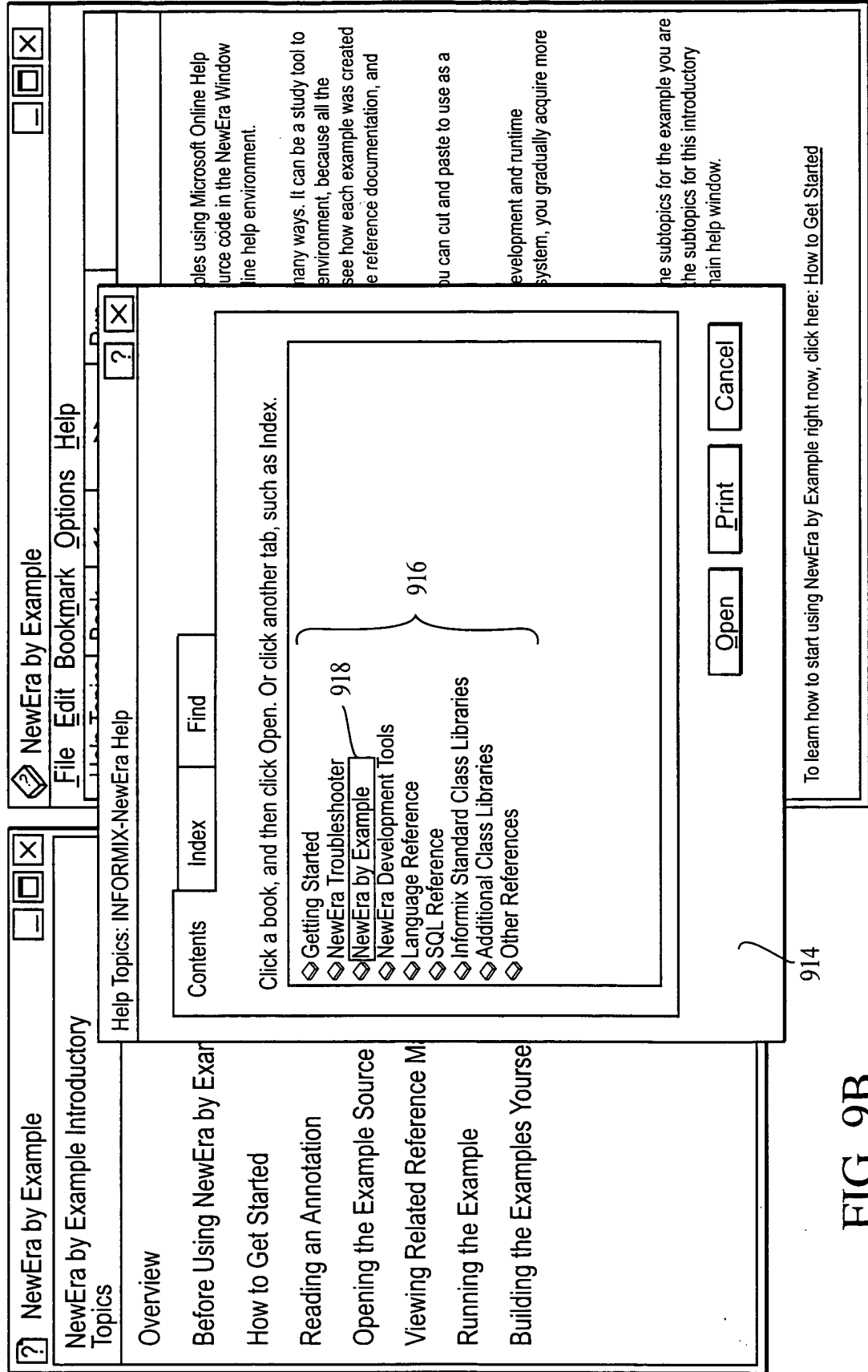


FIG. 9B

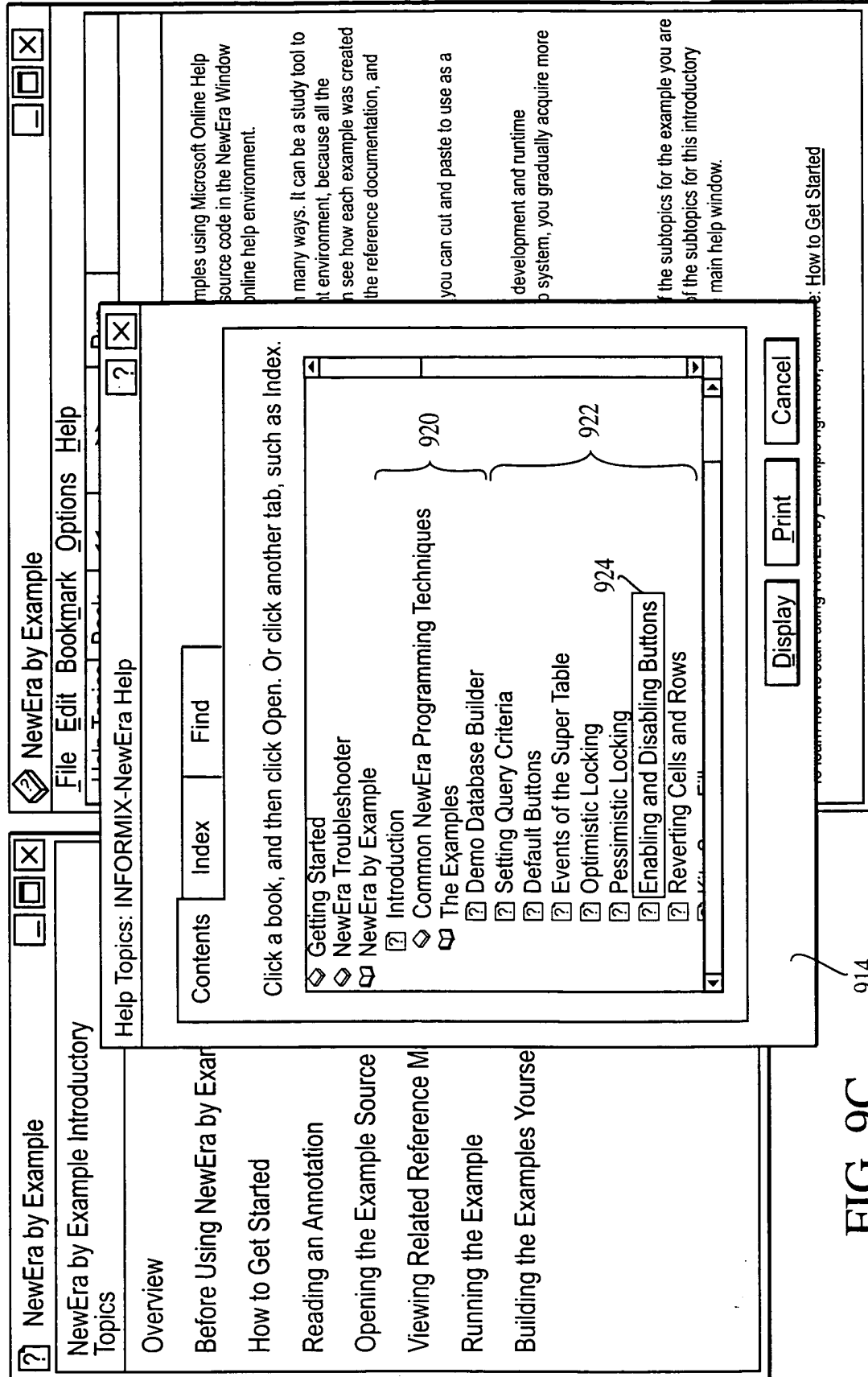


FIG. 9C

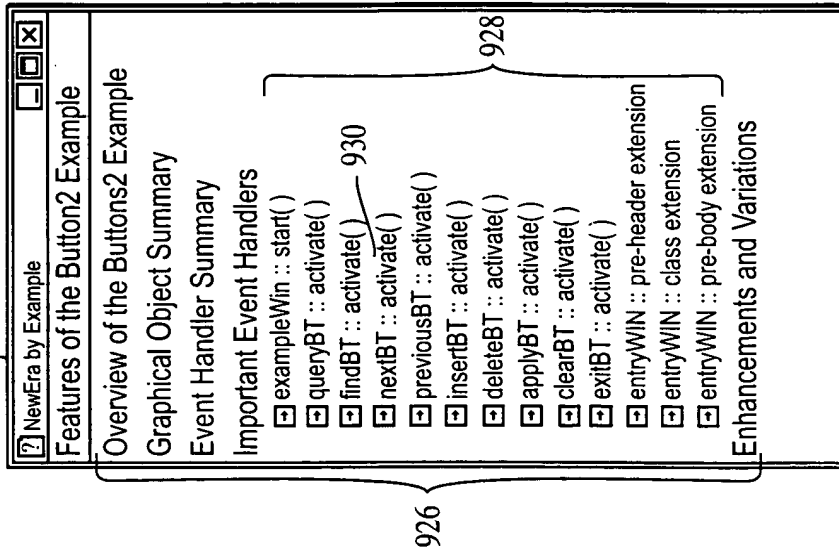
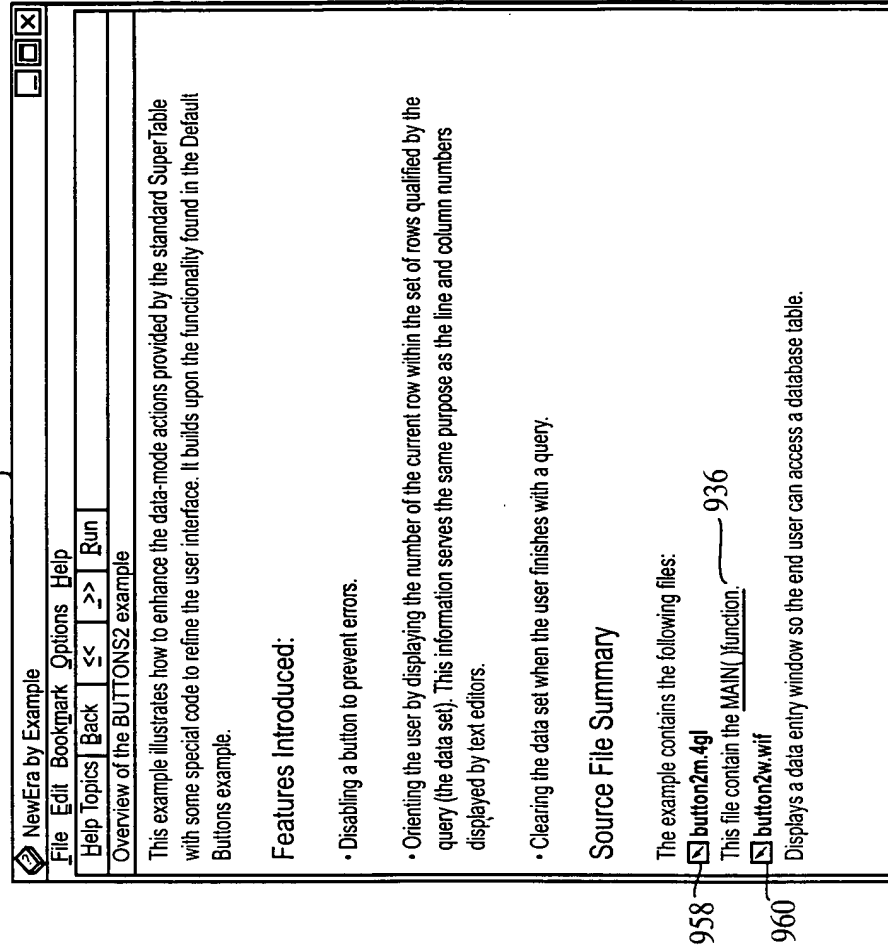
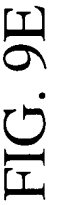


FIG. 9D



904

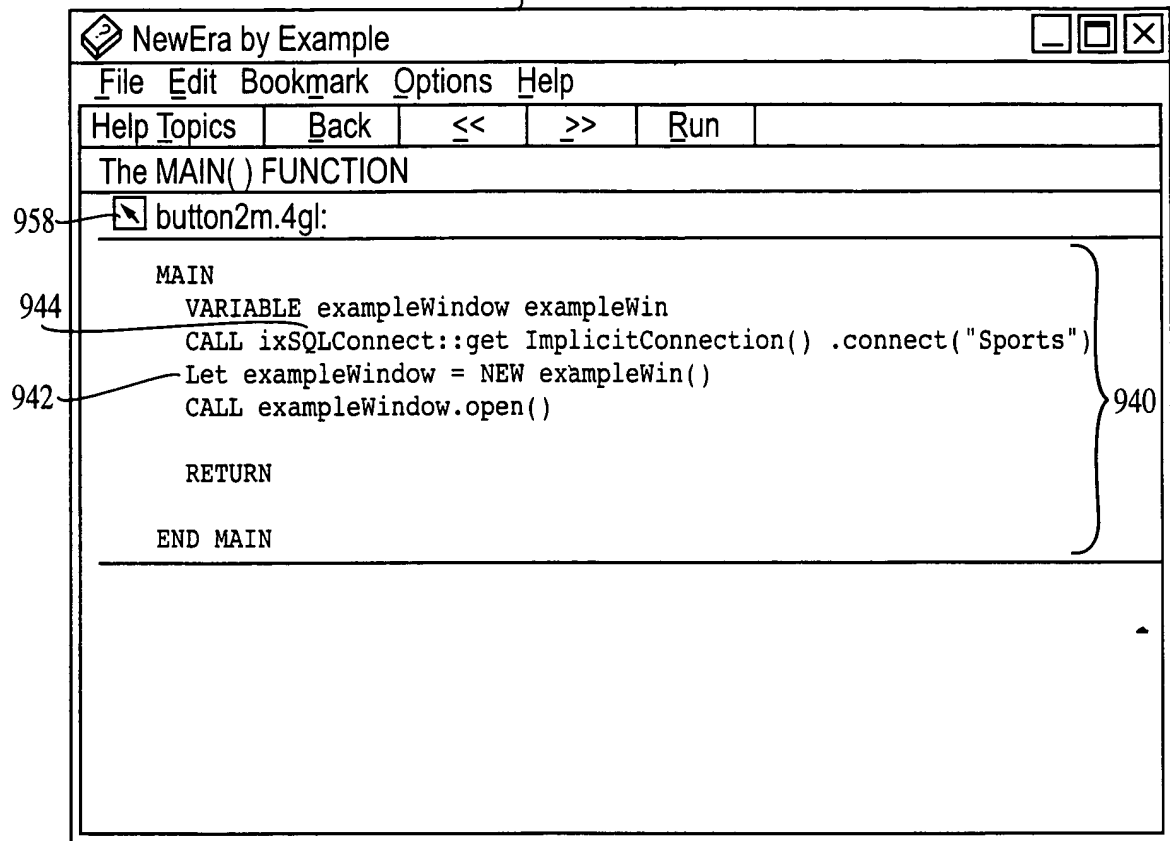


FIG. 9F

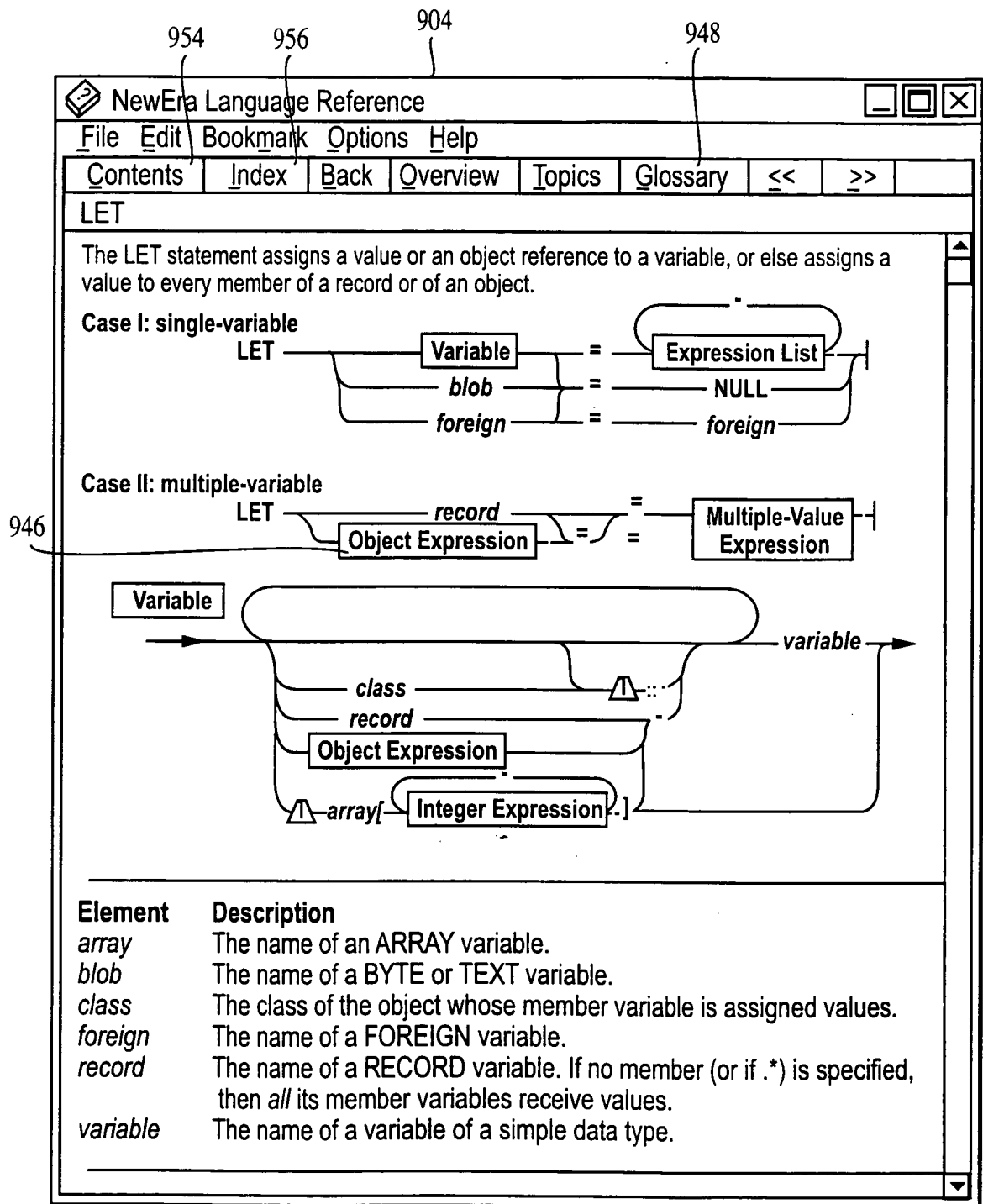


FIG. 9G

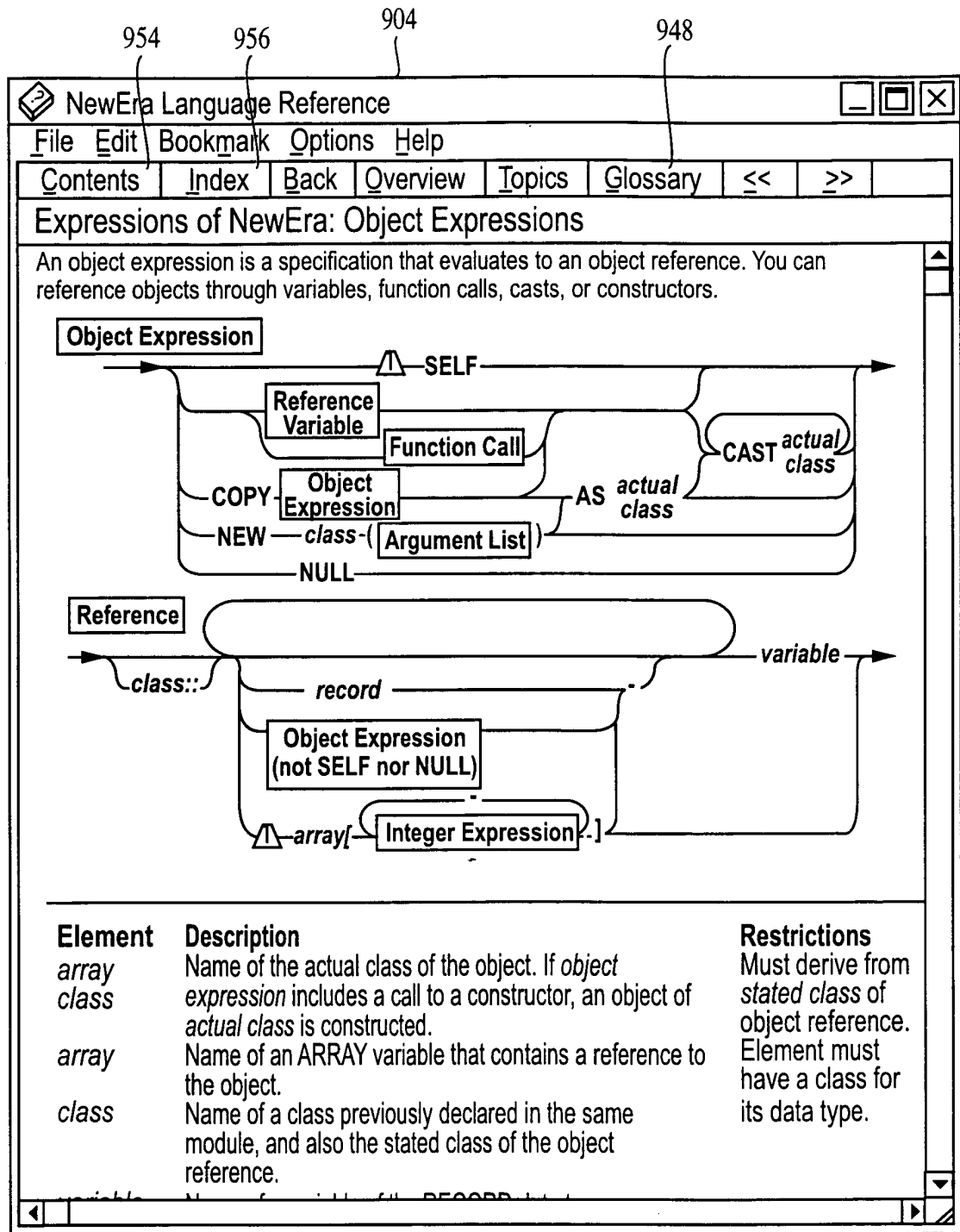


FIG. 9H

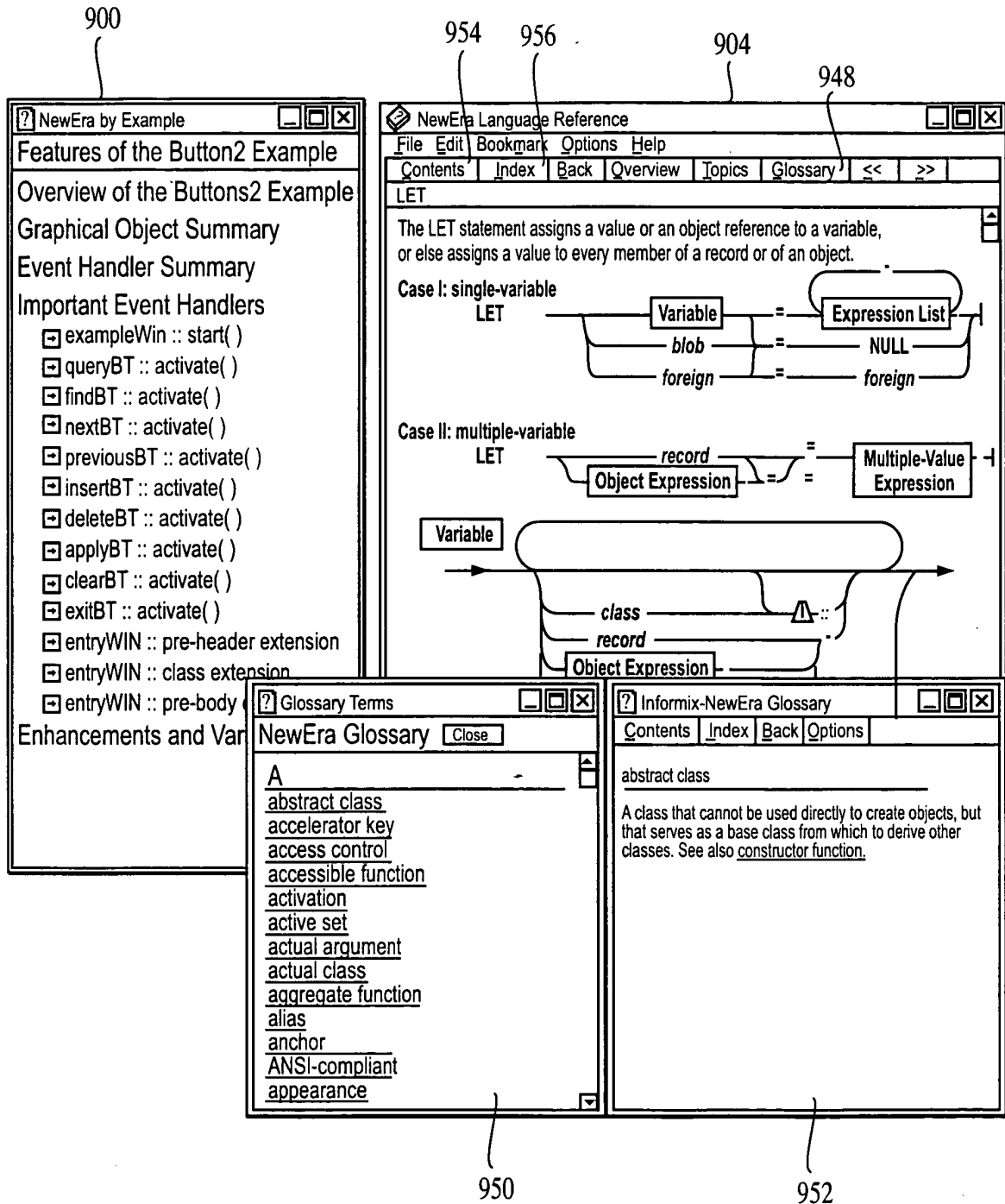


FIG. 9I

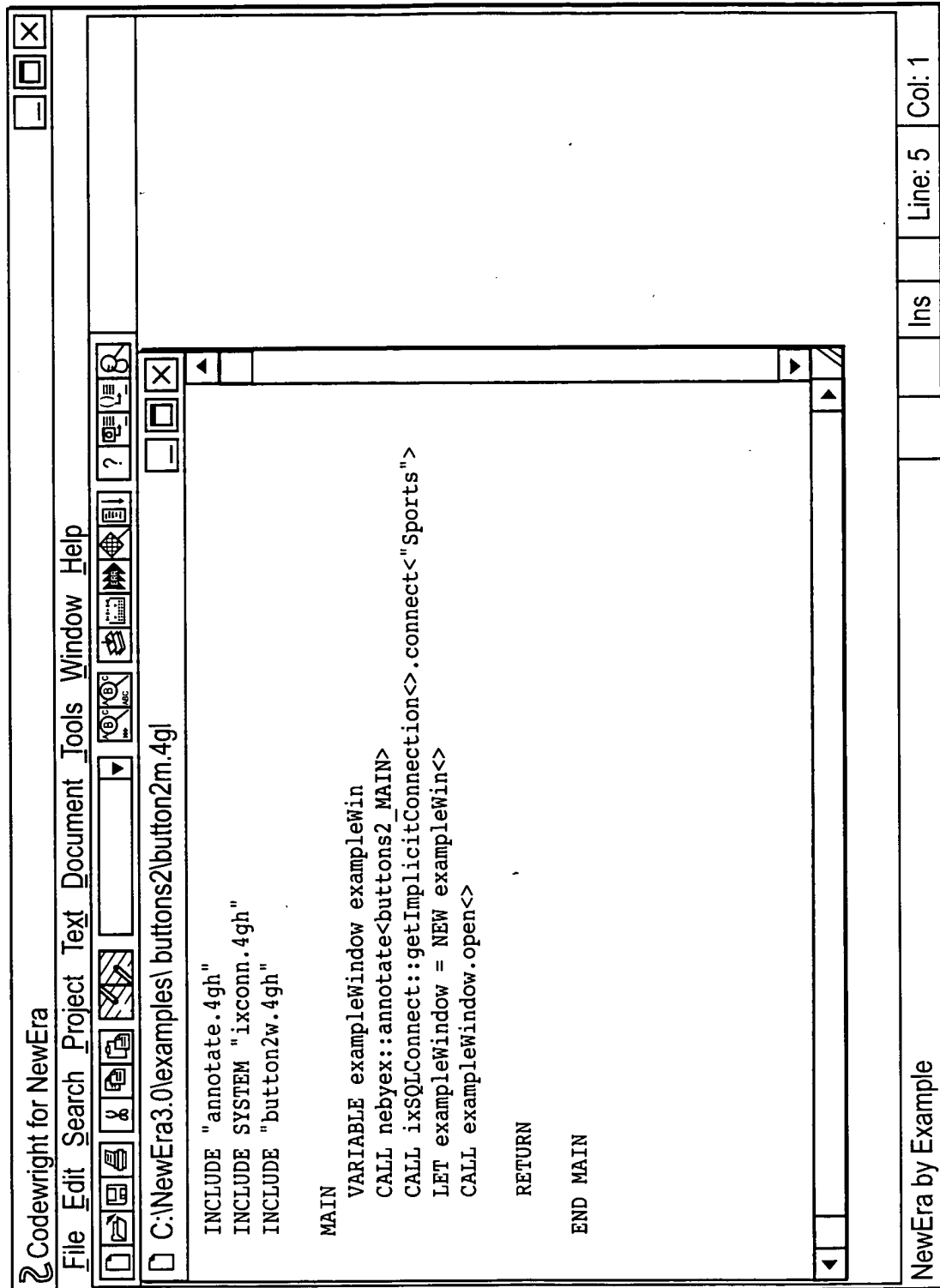


FIG. 9J

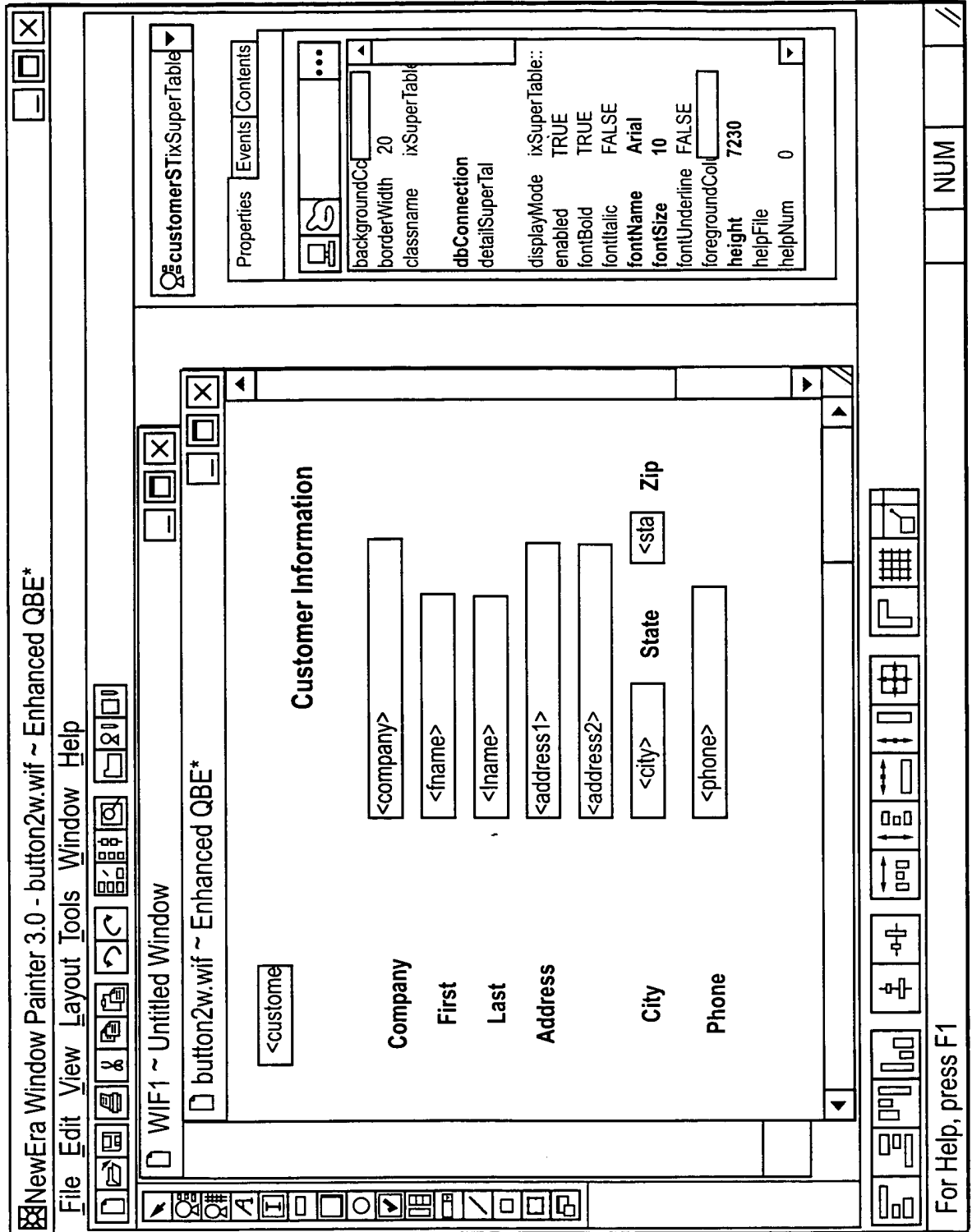


FIG. 9K

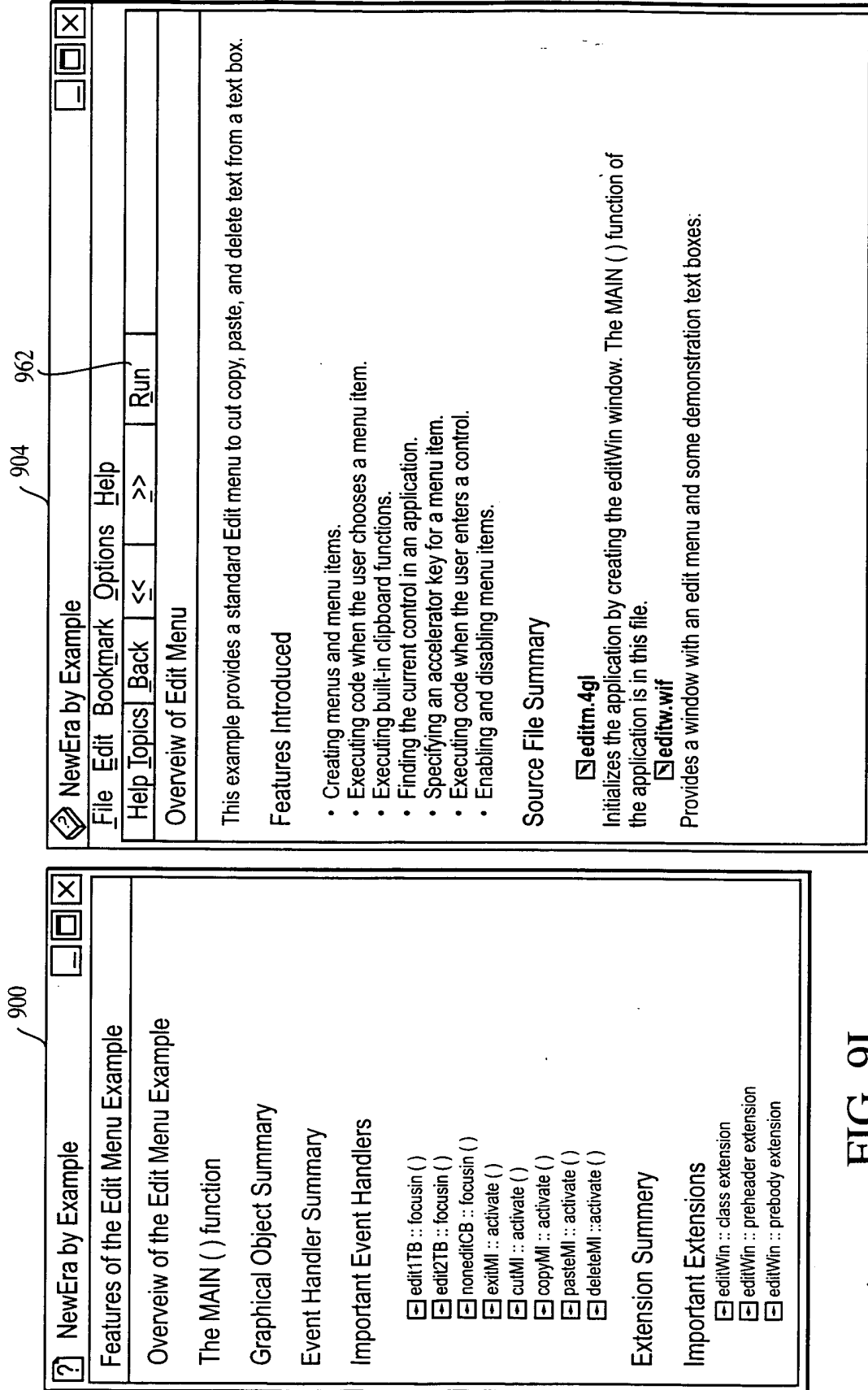
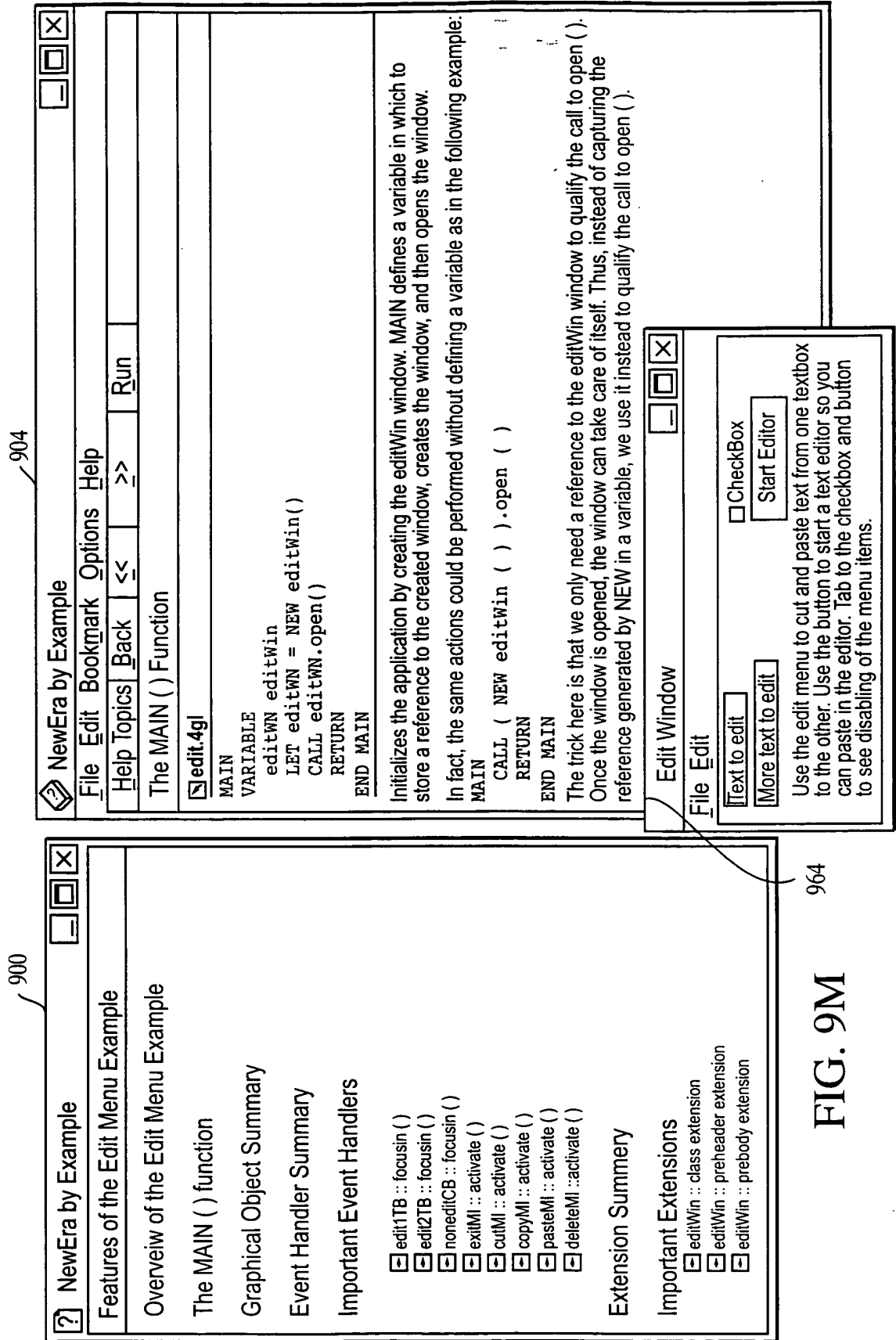


FIG. 9L



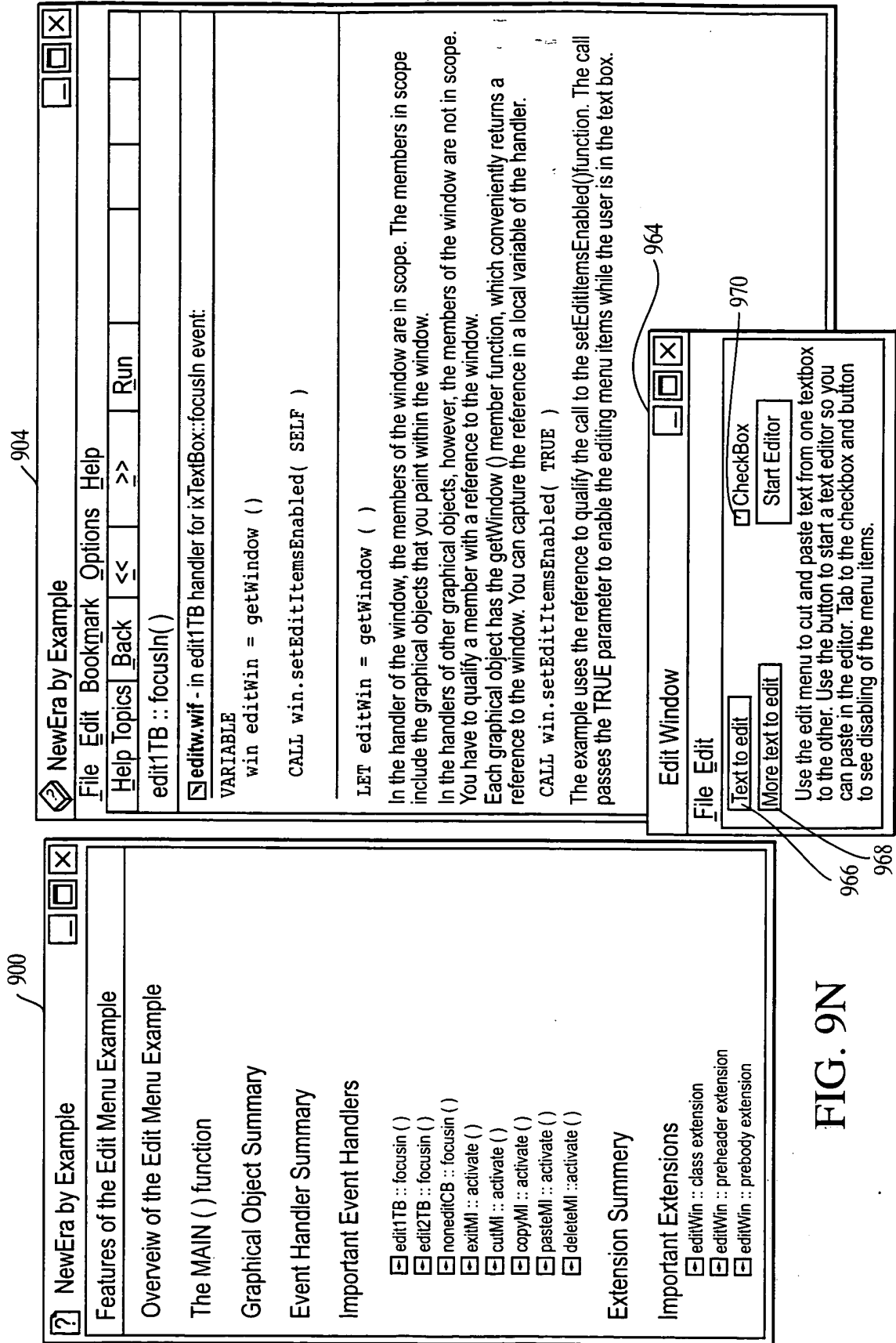
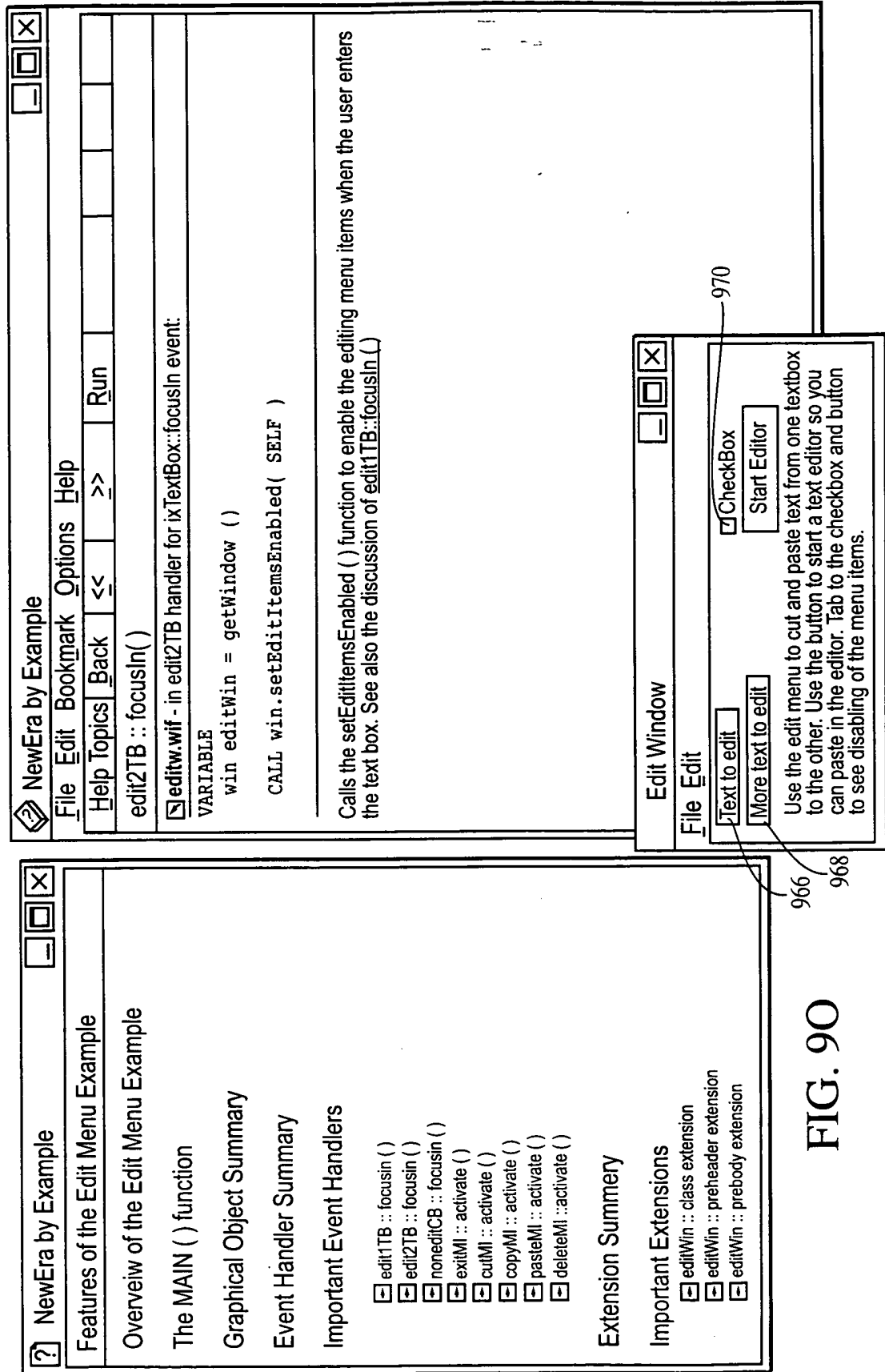


FIG. 9N





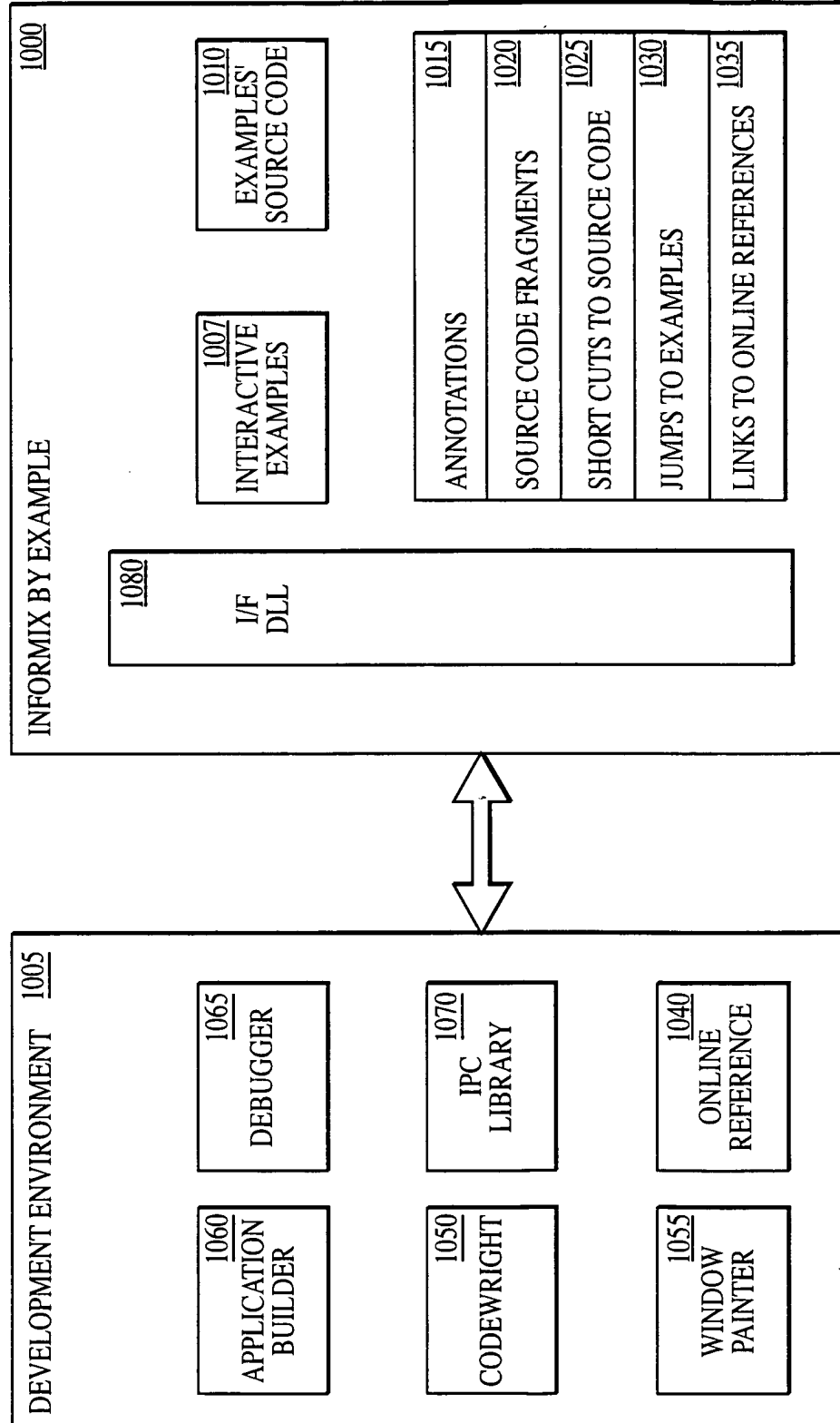


FIG. 10

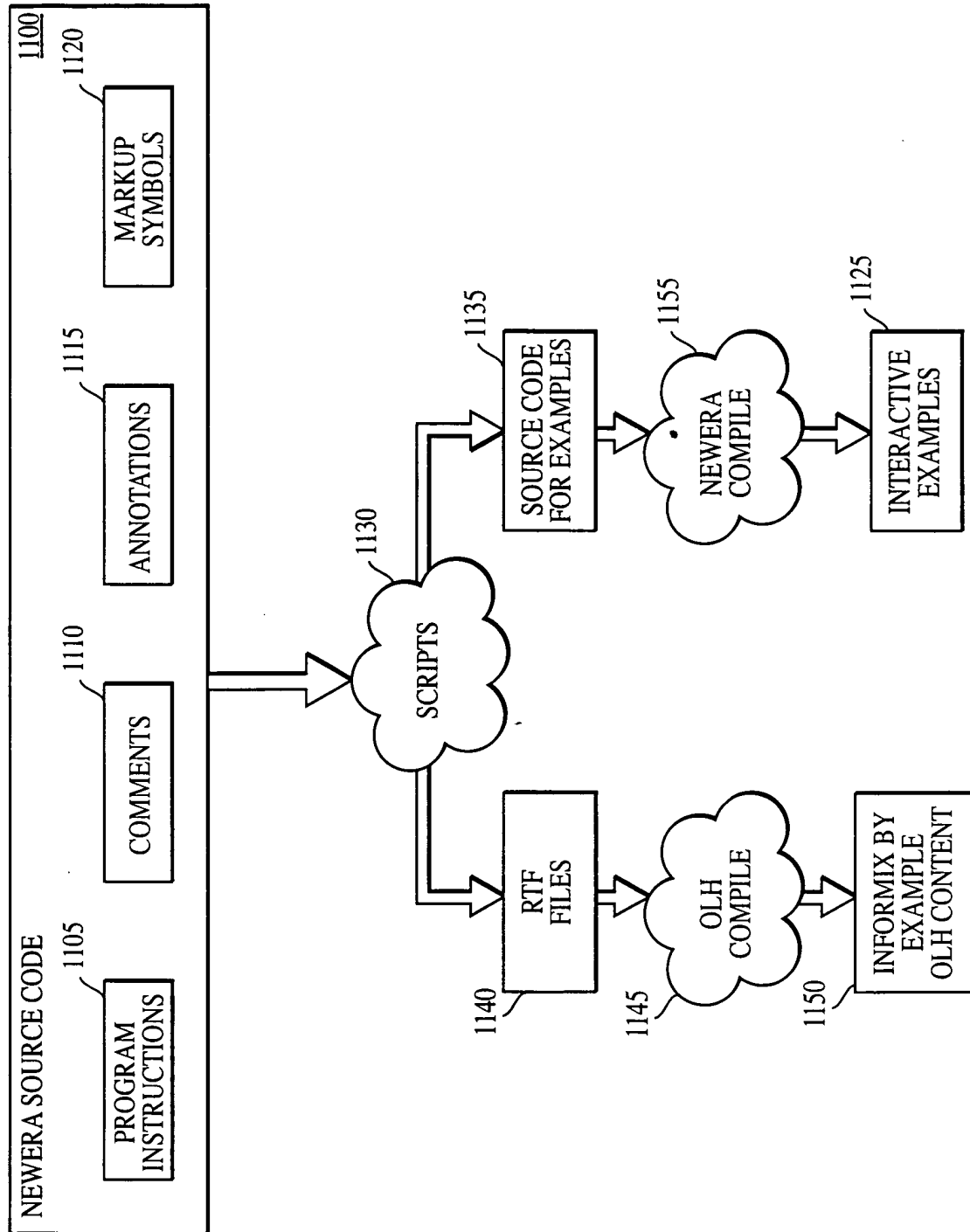


FIG. 11


```
FUNCTION driveStockRpt( destType SMALLINT, destName CHAR(*) )  
1200 RETURNING VOID  
  { .normal  
    Since objects, in particular ixRow objects, cannot be passed  
    as arguments to the report formatter, rows of fetched data will  
    be unpacked into a record that matches the data types and lengths  
    of elements in the fetched rows.  
  }  
  VARIABLE  
    stockRec RECORD  
      mn CHAR(15),      -- manufact.manu_name  
      sn SMALLINT,      -- stock.stock_num  
      sd CHAR(15),      -- stock.description  
      sp MONEY(6,2),    -- stock.unit_price  
      su CHAR(4)        -- stock.unit  
    END RECORD  
  
    stockStmt ixSQLStmt,  
    stmtString CHAR(*),  
    stockRow ixRow,  
  
    errorCode INTEGER,  
    logFile ixErrorLog  
1205  { .normal  
    Use the implicit connection object to create an SQL statement  
    object. The connection object must already be connected to a  
    database.  
    Checking the status of the prepare( ) call will confirm this.  
1210 }  
    { .[edit stmt]  
      LET stockStmt =  
        ixSQLConnect::getImplicitConnection().createStmtObject()  
    { .]file stmt}  
1215
```

FIG. 12

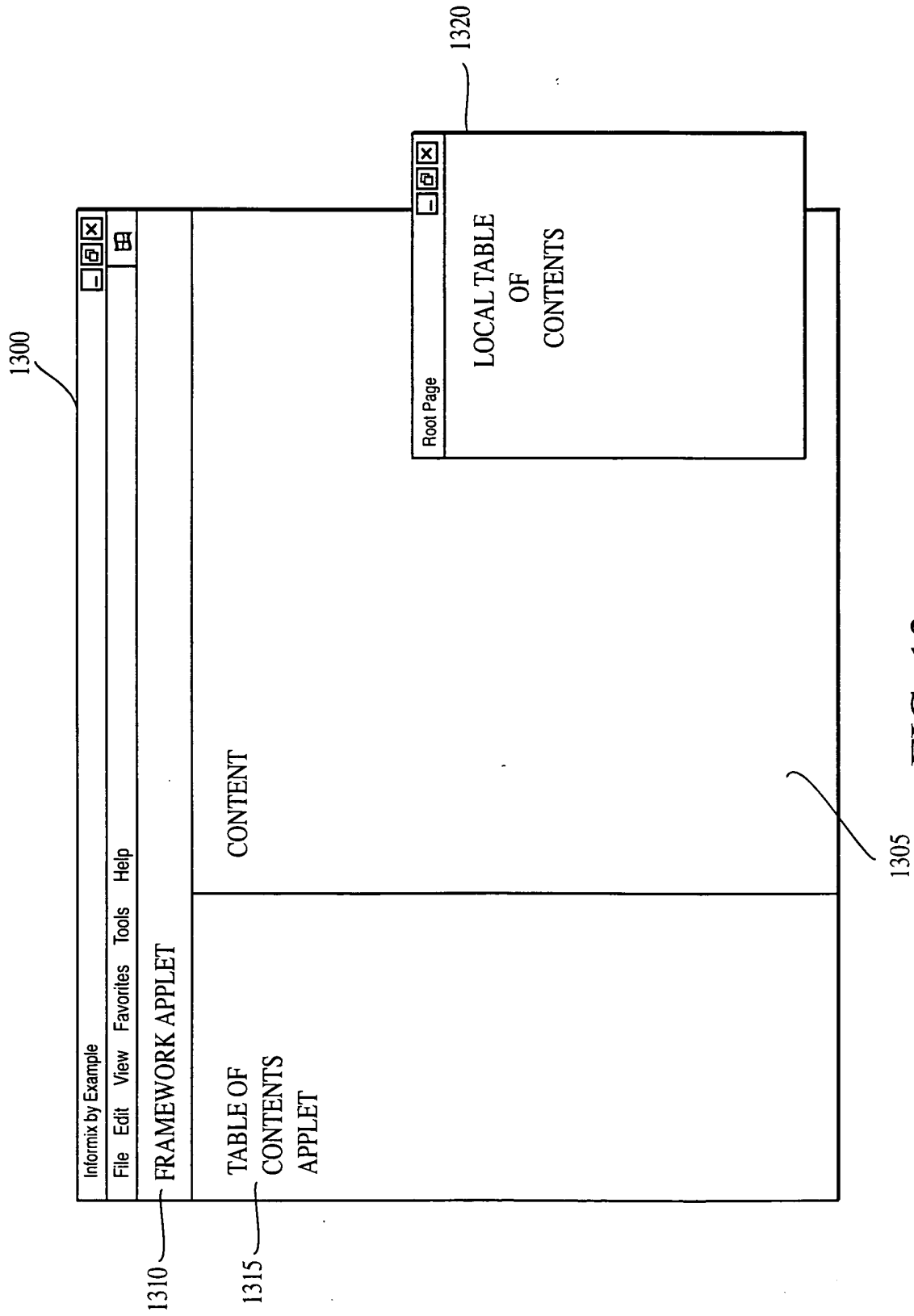


FIG. 13

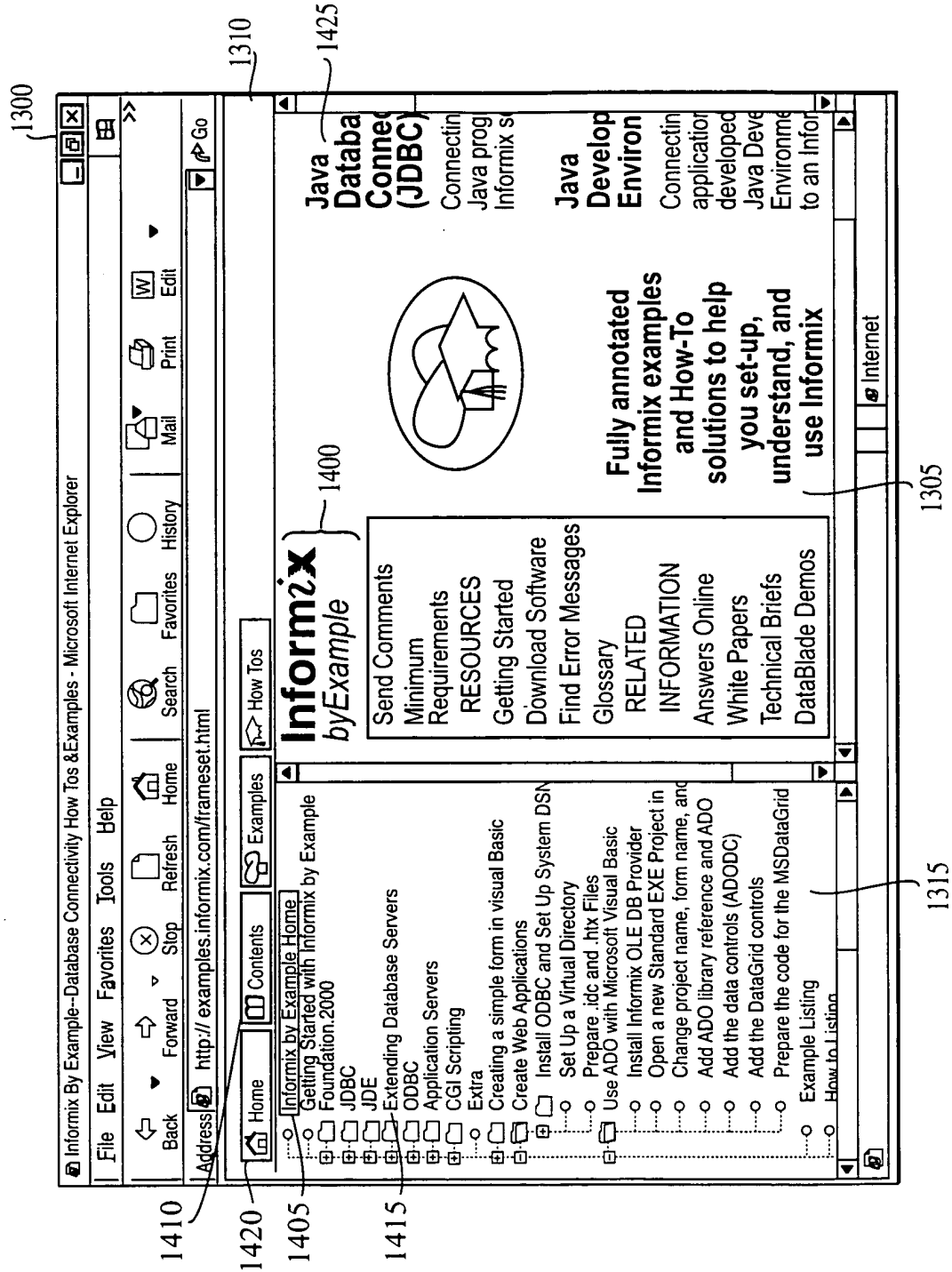


FIG. 14A

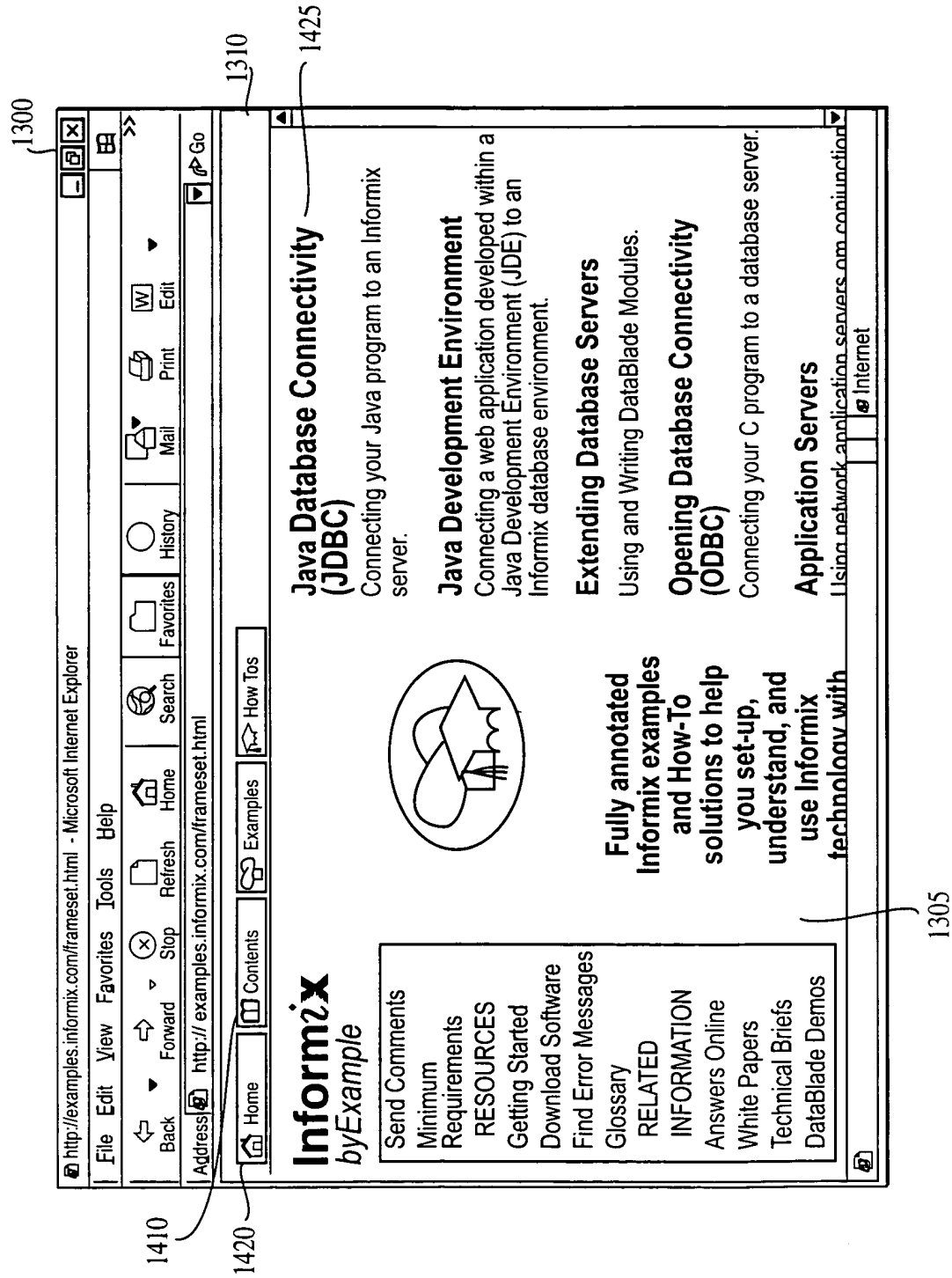


FIG. 14B

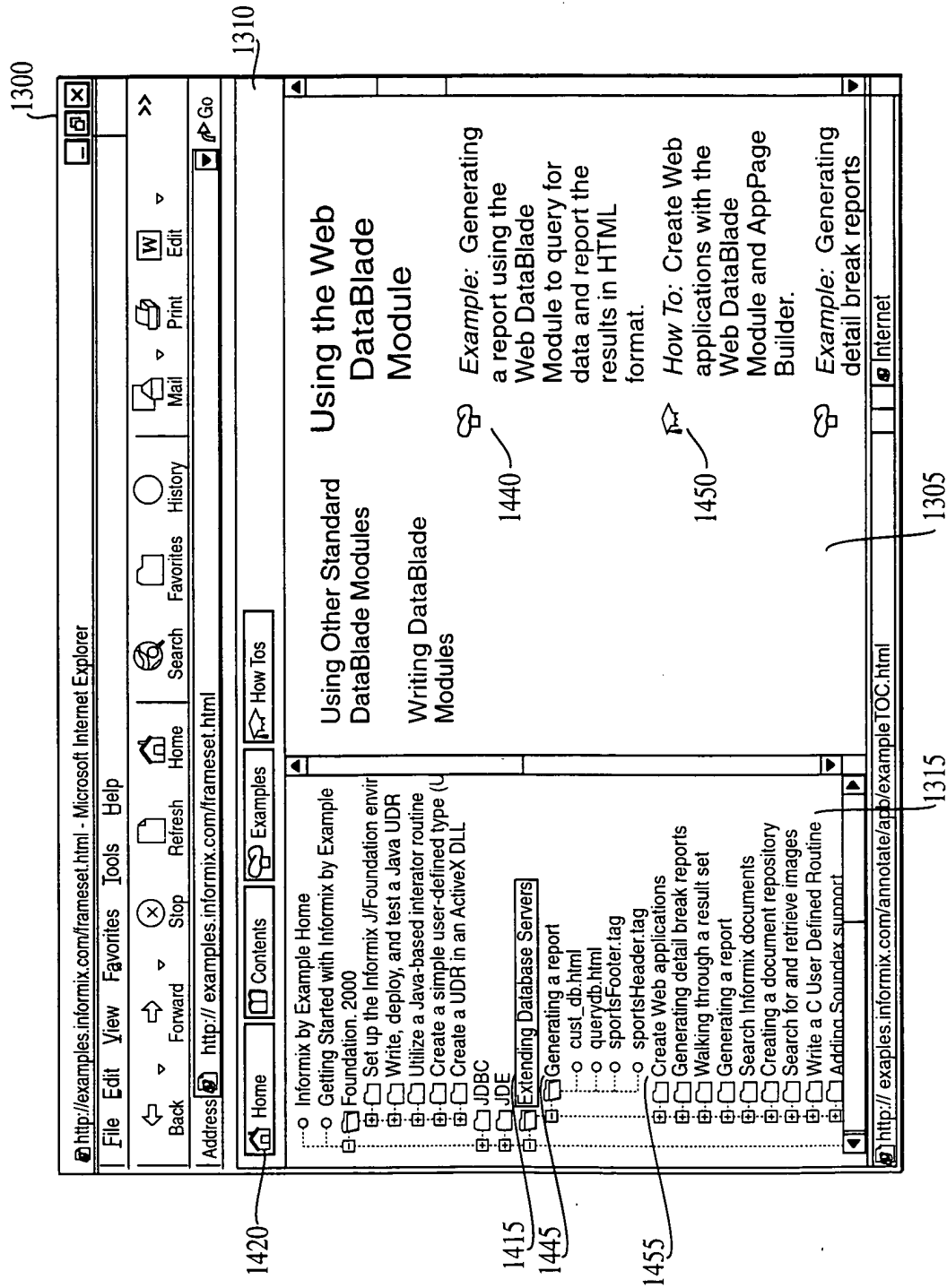


FIG. 14C

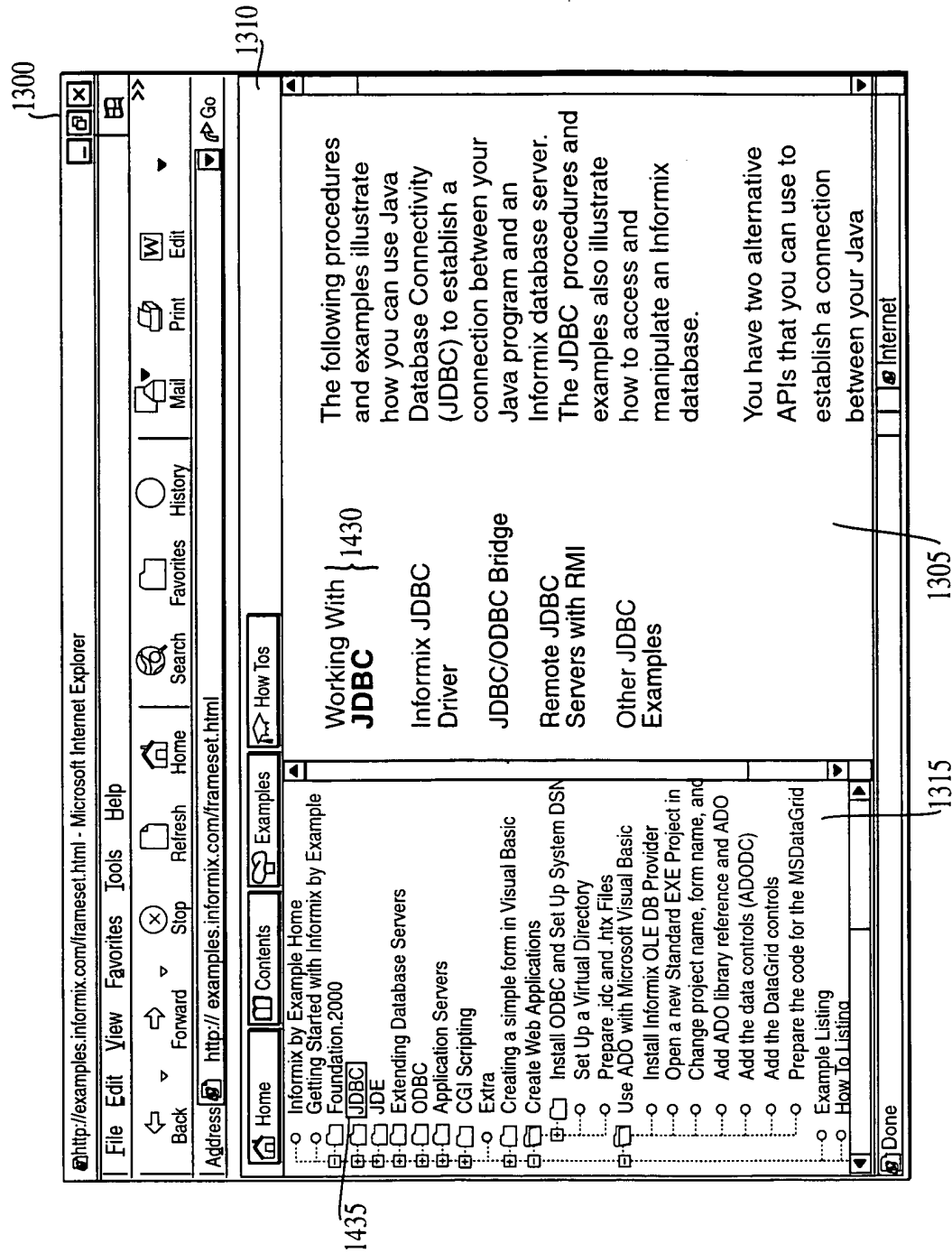


FIG. 14D

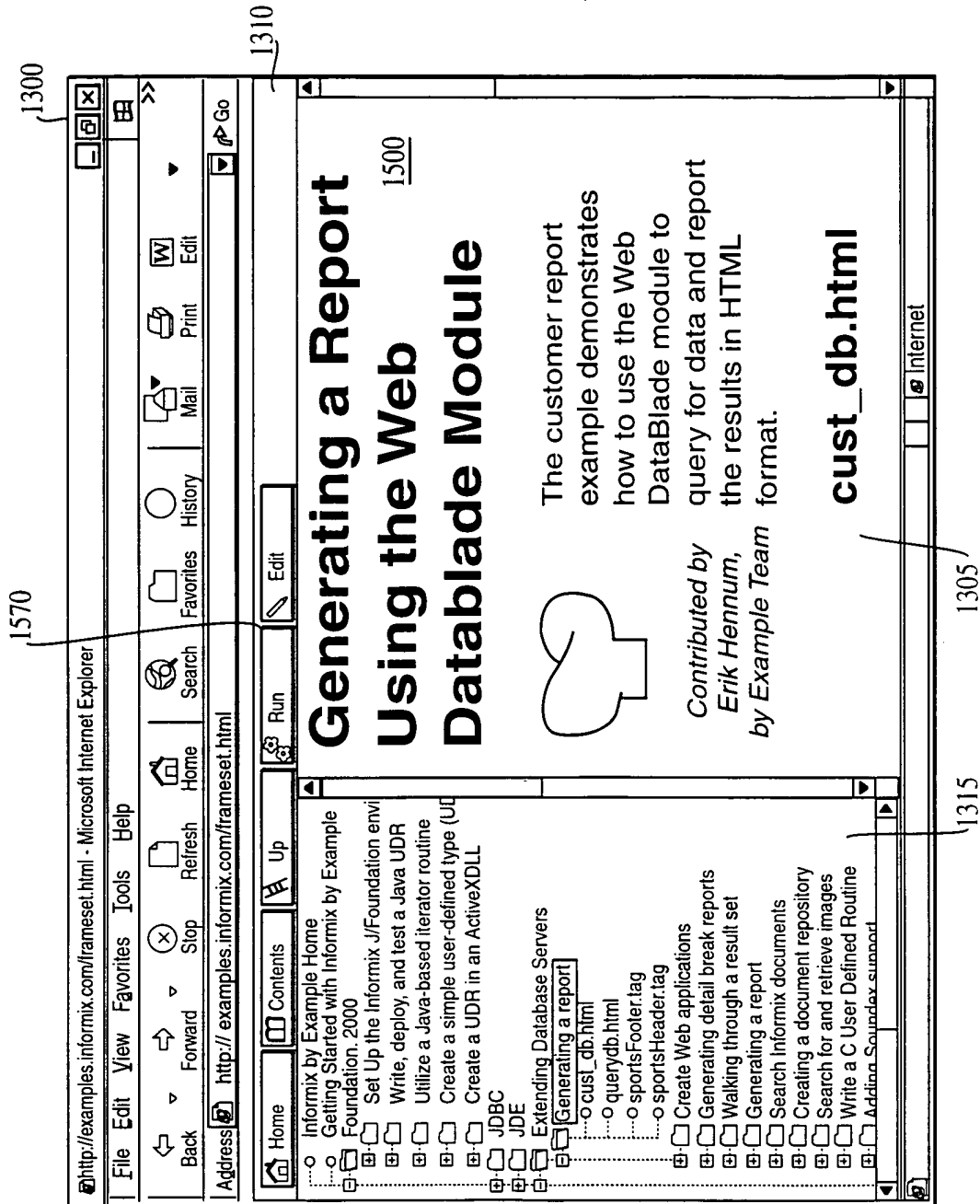
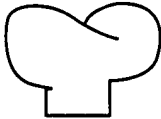


FIG. 15A

1500

Generating a Report Using the Web Datablade Module



The customer report example demonstrates how to use the Web DataBlade module to query for data and report the results in HTML format. — 1505

*Contributed by
Erik Hennum,
byExample Team*

`cust_db.html` — 1510

1505

This app page accepts a query and generates an HTML report

1510

`querydb.html` — 1510

1505

This HTML page contains a form that invokes an app page

`sportsFooter.tag` — 1510

1505

The sportsFooter dynamic tag generates the footer for an app page.

`sportsHeader.tag` — 1510

1505

The sportsHeader dynamic tag generates the header for an app page.


 Click here to view or print all of the source files for this example.

FIG. 15B

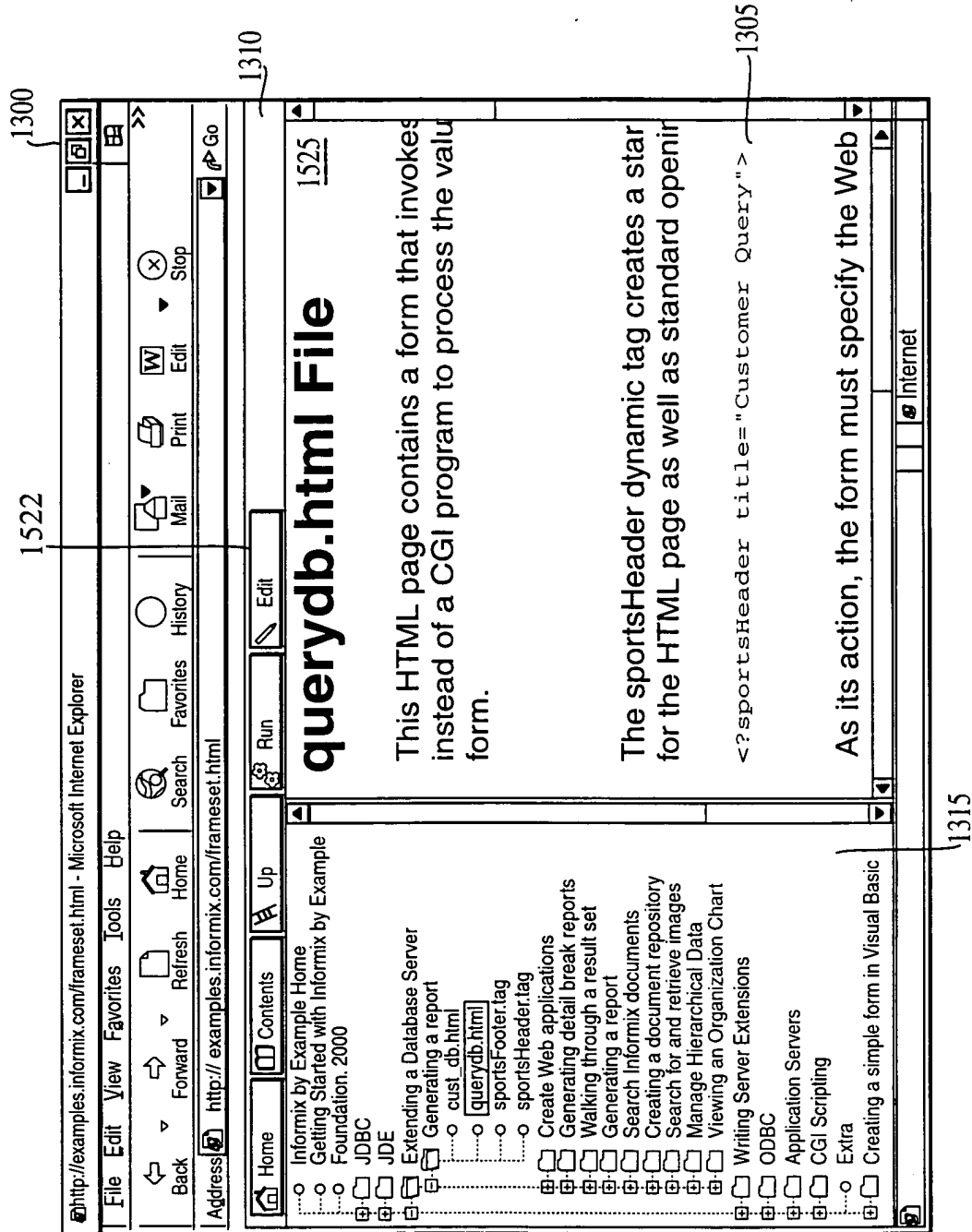


FIG. 15C

1525

querydb.html File

This HTML page contains a form that invokes an app page instead of a CGI program to process the values in the form.

The sportsHeader dynamic tag creates a standard header for the HTML page as well as standard opening text.

1535

<?sportsHeader title="Customer Query">

1530

1540

As its action, the form must specify the Web Driver utility.

<P>

<FORM ACTION = "<?MIVAR>\$WEB_HOME<?/MIVAR"METHOD="GET">

1530

1550

To specify the app page, the form must use a hidden input component. The input component must have a name of **Mival** and a value that's the name of the app page. The input component below specifies the cust_db.html app page.

1535

<INPUT TYPE="HIDDEN" NAME="Mival" VALUE="/examples/CustRpt/cust_db.html">

Optional state:

<INPUT TYPE="TEXT" NAME="SelectState" SIZE="3" MAXLENGTH="2">

<INPUT TYPE="SUBMIT" NAME="SUBMIT" VALUE="SUBMIT">

</FORM>

</P>

1530

</BODY>

</HTML>

FIG. 15D

querydb.html

1560

```
<!--<ibyx>
<intro>
<p><abstract>This HTML page contains a form that invokes
an app page</abstract> instead of a CGI program to process
the values in the form.
</p>
</intro>
</ibyx> -- >

<!-- <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> -->
<?sportsHeader title="Customer Query">

<!-- <ibyx>
<p> As its action, the form must specify the Web Driver utility.
</p>
</ibyx>-->
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR>" METHOD="GET">

<!-- <ibyx>
<p> To specify the app page, the form must use a hidden input component.
The input component must have a name of <strong>Mival</strong> and
a value that's the name of the app page. The input component below
specifies the <a href="cust_db.html">cust_db.html</a> app page.
</p>
</ibyx> -->
<INPUT TYPE="HIDDEN" NAME="Mival" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPE="TEXT" NAME="selectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">

</FORM>
</P>

<?annotate>

</BODY>
</HTML>
```

FIG. 15E

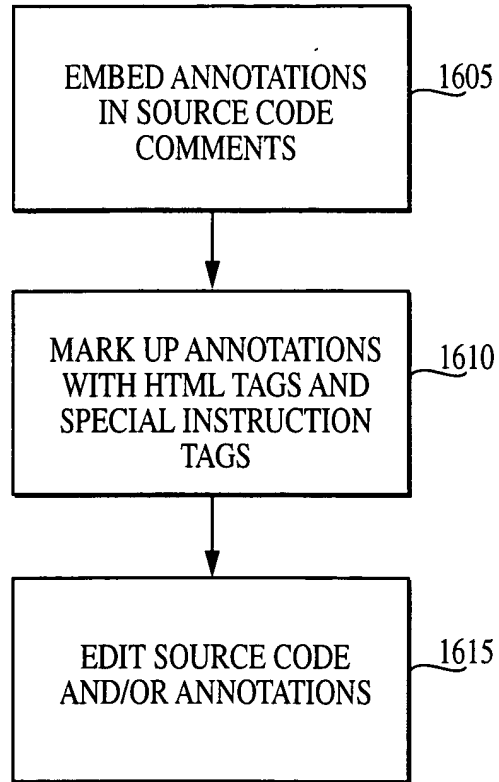


FIG. 16A

1620

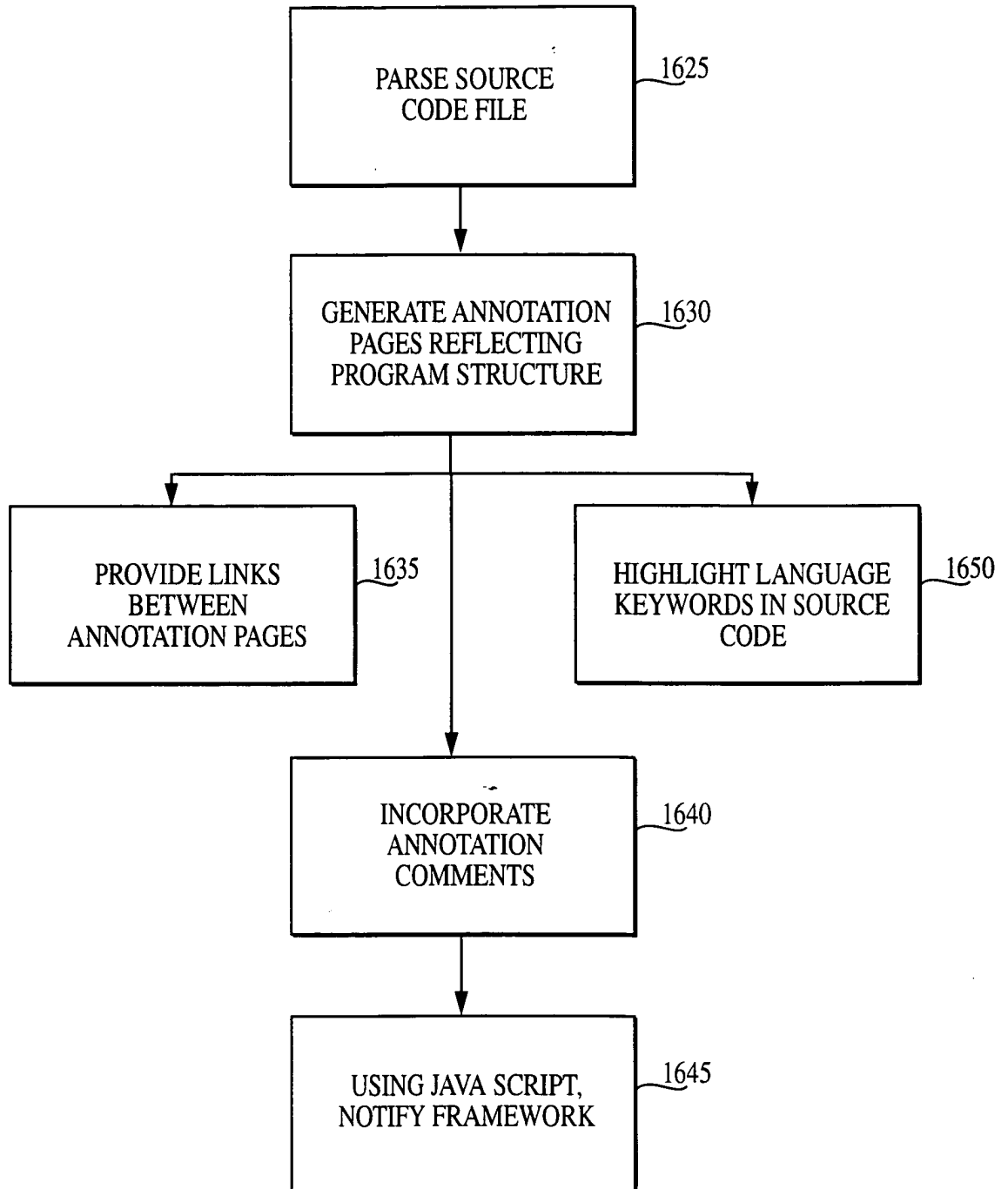


FIG. 16B

1655

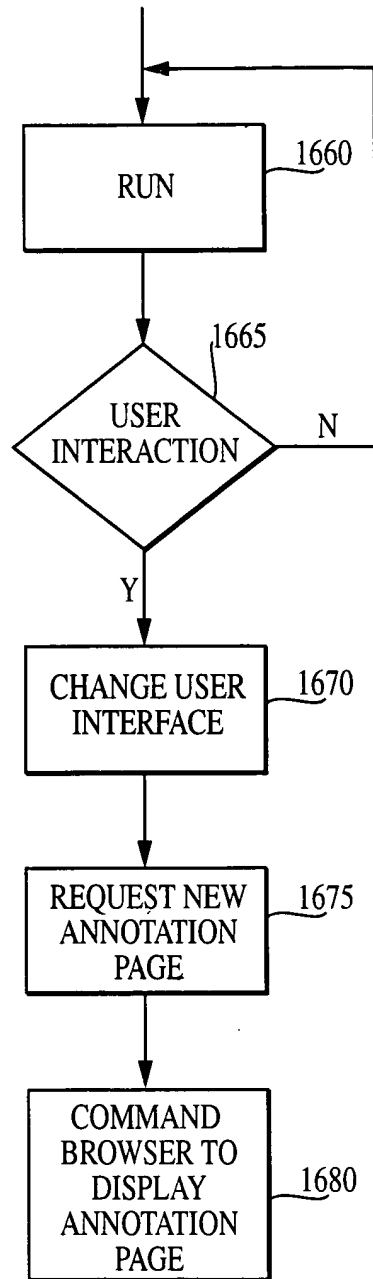


FIG. 16C

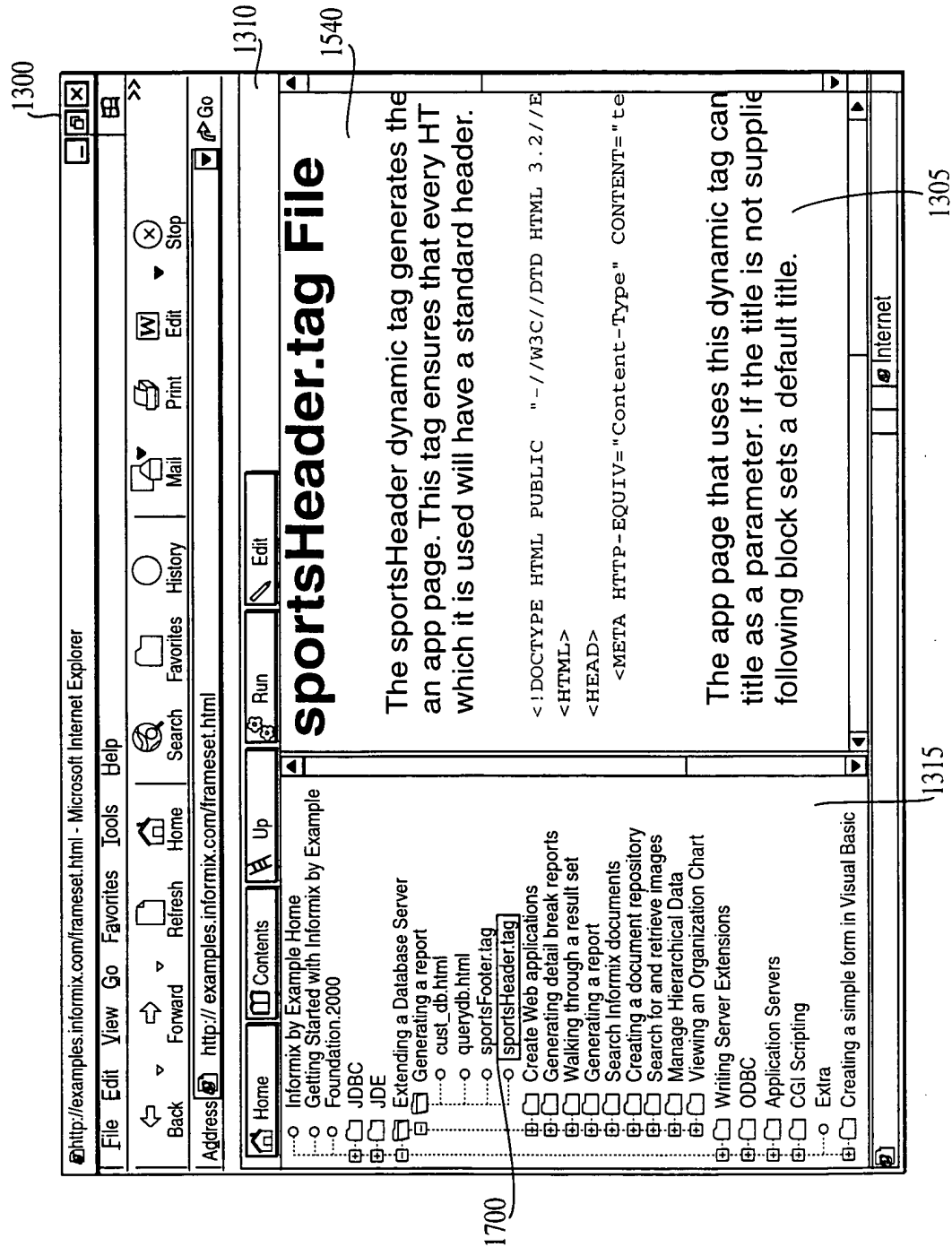


FIG. 17A

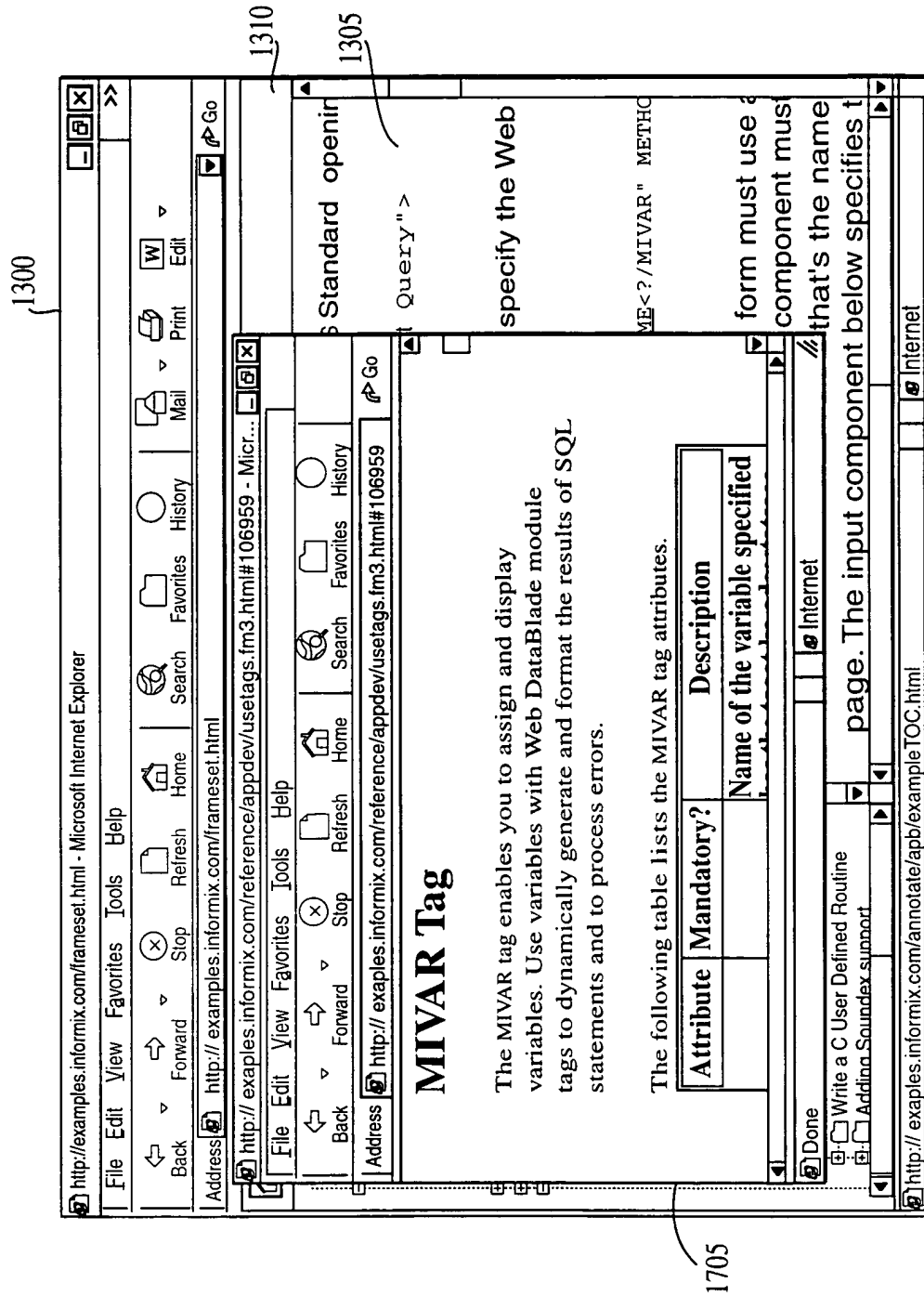


FIG. 17B

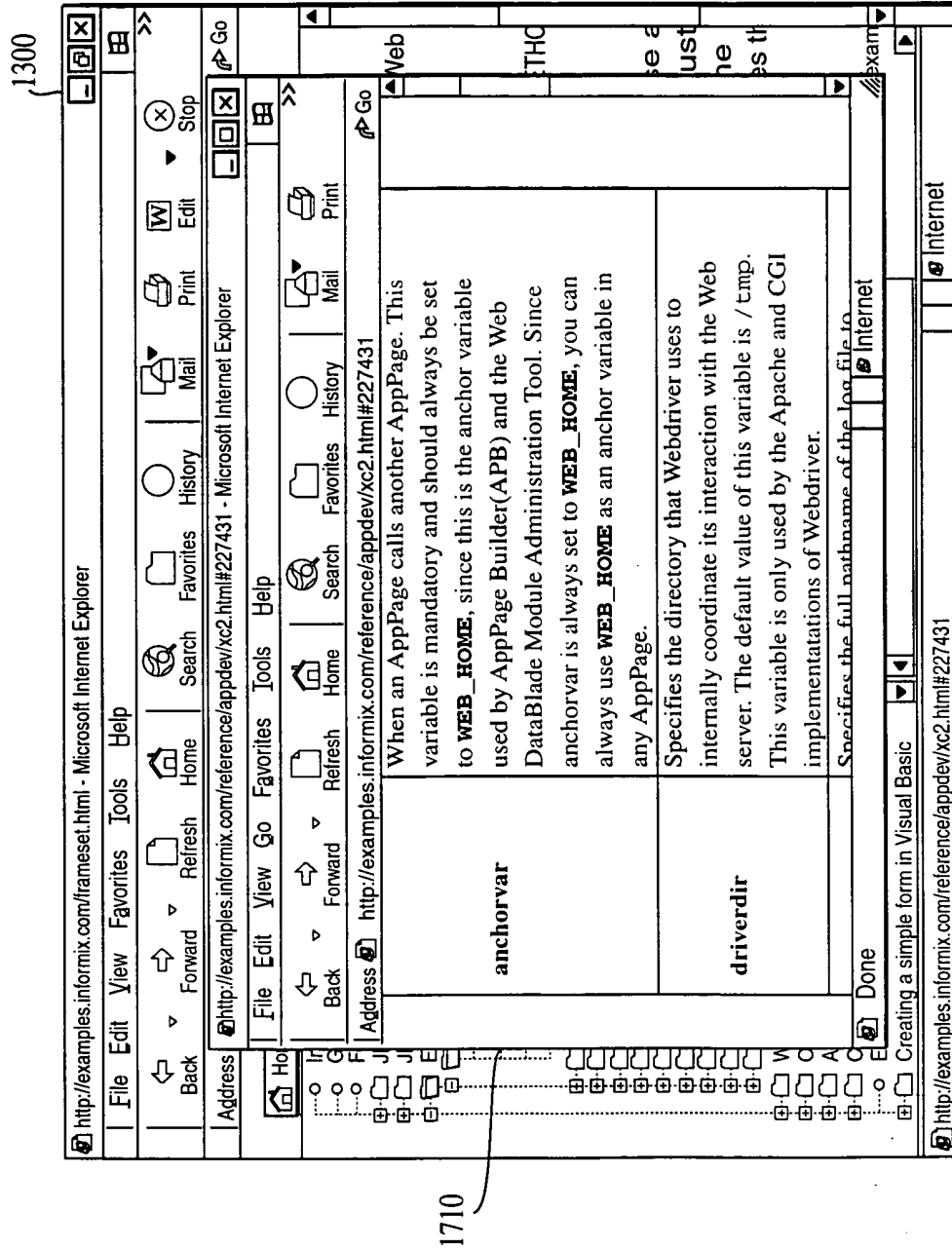


FIG. 17C

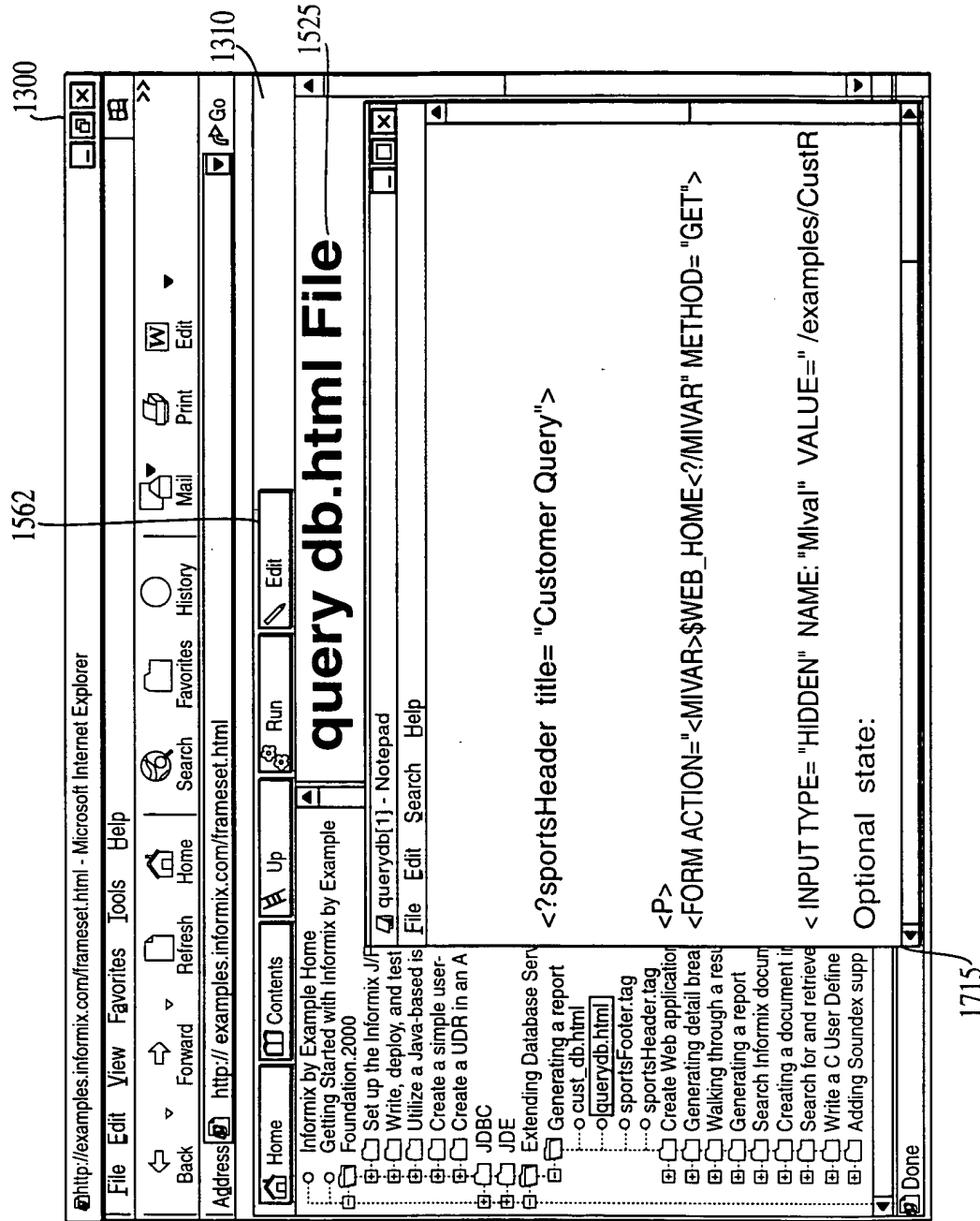


FIG. 17D

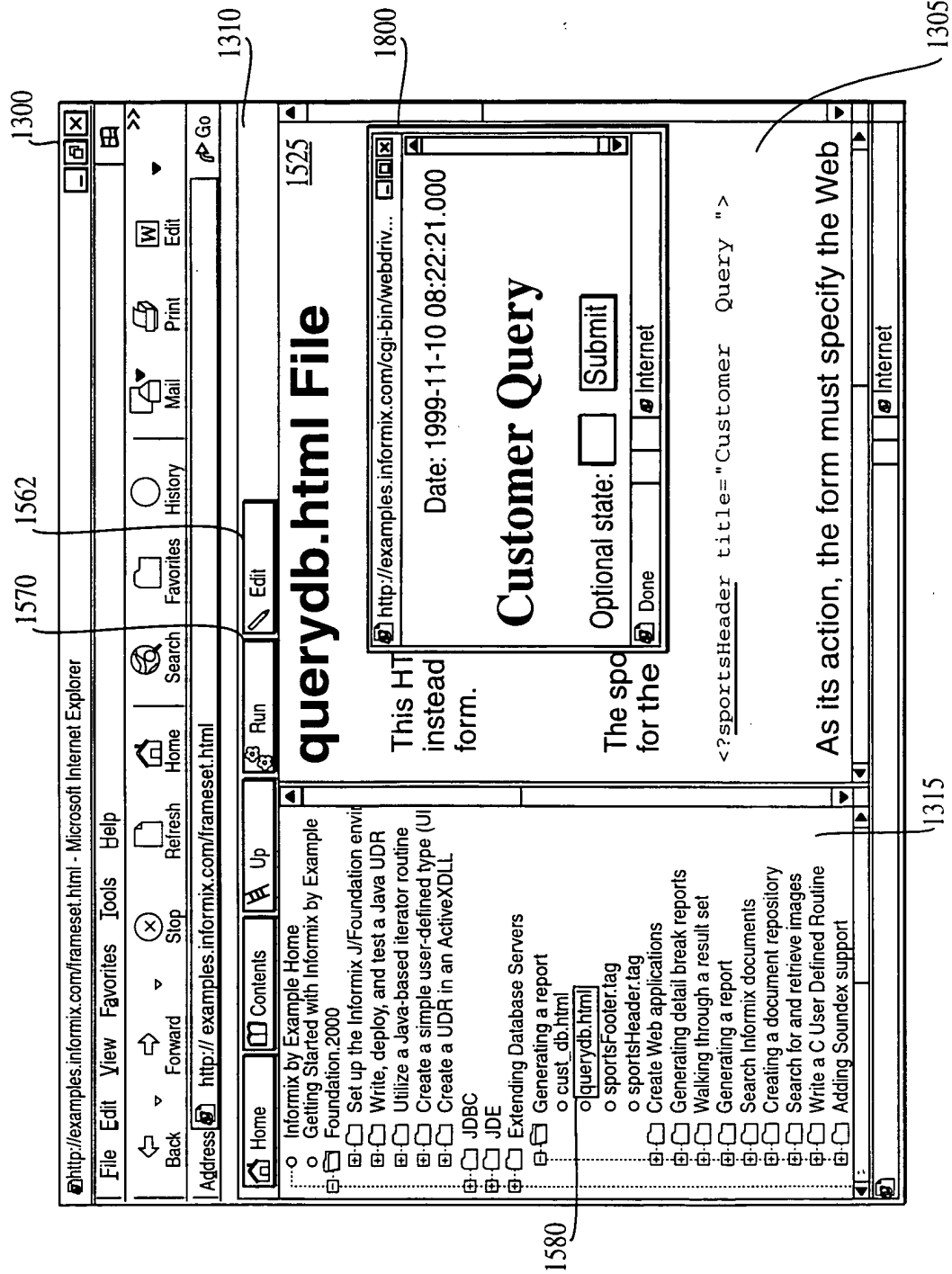


FIG. 18A

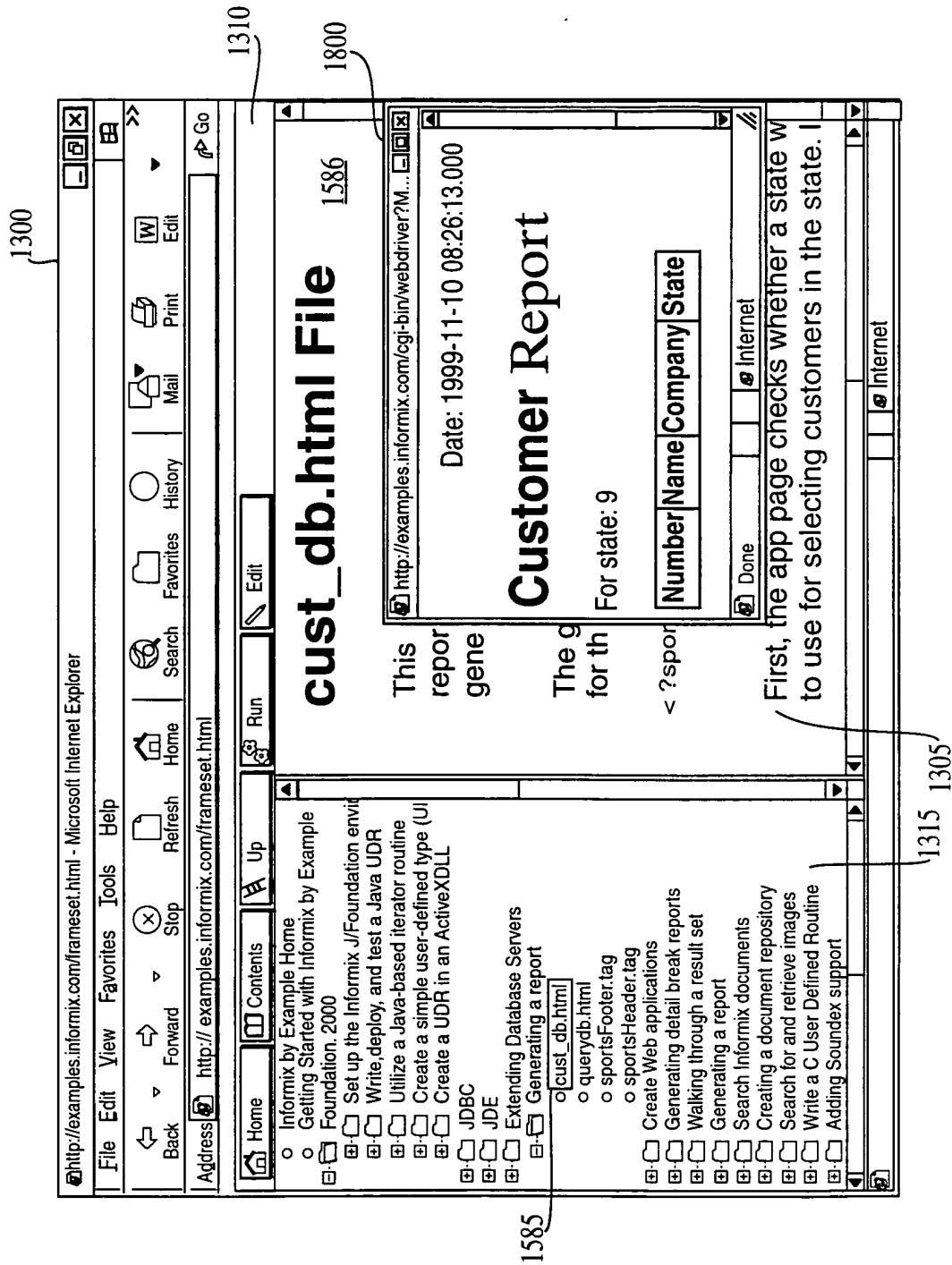


FIG. 18B

1810 {

```

<! - - <ibyx>
<intro>
<p> <abstract>This app page accepts a query and generates an HTML
report</abstract> in response.
The app page uses dynamic tags to generate the header and footer for the
HTML report.
</p>
</intro>
</ibyx> - - >
<! - - <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> - ->
<?sportsHeader title="Customer Report">

<! - - <ibyx>
<p> First, the app page checks whether a state was specified to use for
selecting customers in the state. If so, the block generates a
paragraph to identify the state.
</p>
</ibyx> - - >
<?MIVAR NAME=$WHERE_STR></MIVAR>
<?MIBLOCK COND="$(AND,$(XST,$selectState) , $ (<,0,$ (STRLEN, $selectState)))">
    <?MIVAR NAME= $WHERE_STR>WHERE state= " $selectState"</MIVAR>
    <?MIVAR><P>For state: $selectState</P></MIVAR>
</MIBLOCK>

<! - - <ibyx>
<p>Next, the app page starts the table that will contain the data.
</p>
</ibyx> - ->
    <P><TABLE BORDER="1">
        <TR>
            <TH>Number</TH><TH>Name</TH><TH>Company</TH><TH>State</TH>
        </TR>

<! - - <ibyx>
<p> The MISQL block queries for customers, optionally selecting only customers
from the specified state. Because the contents of the block are generated
for every row of data, a new table row describes each customer.

The &nbsp; HTML entity is a non-breaking space. By putting a non-breaking
space in each column, we force the Web Browser to display the column even
if the value is null.
</p>
</ibyx> - ->
<?MISQL SQL= "SELECT customer_num, fname, lname, company, state FROM customer $WHERE_STR; ">
    <TR>
        <TD> $1&nbsp; </TD><TD>$2&nbsp; <TD>$3</TD><TD>$4&nbsp;</TD><TD>$5&nbsp;</TD>
    </TR>
</MISQL>

    </TABLE></P>

< ! - - <ibyx>
<p> The sportsFooter dynamic tag creates a standard footer
for the HTML page.
</p>
</ibyx> - ->
<?sportsFooter>

```

FIG. 18C

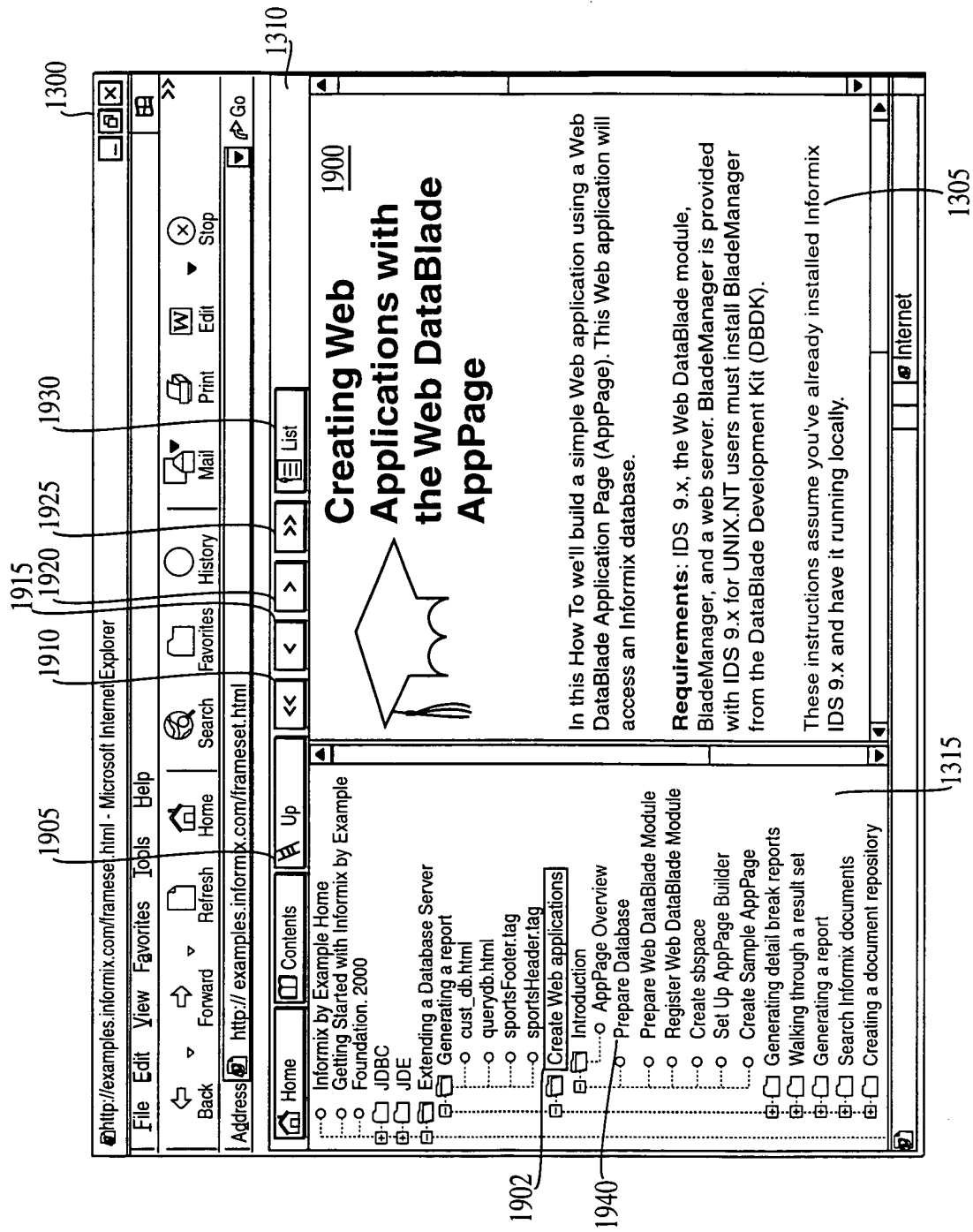
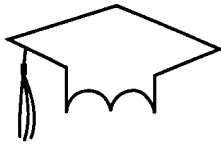


FIG. 19A

1900



Creating Web Applications with the Web DataBlade AppPage

1930

In this How To we'll build a simple Web application using Web DataBlade Application Page (AppPage). This Web application will access an Informix database.

Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX. NT users must install BladeManager from the DataBlade Development Kit (DBDK)

These instructions assume you've already installed Informix IDS 9.x and have it running locally.

1935

Define a server connection and prepare a sample database.

1. Define a server connection with setnet32 (NT). Create a sample database or use the stores7 demo database. ▶ Prepare Database.

Prepare the Web DataBlade development environment.

2. Install the Web DataBlade module and BladeManager. ▶ Prepare Web DataBlade Development Environment.

3. Register the Web DataBlade module in the demo database with BladeManager. ▶ Register the Web DataBlade.

4. Create a sbspace for smart large objects, like gifs. ▶ Create Smart Blob Space (sbspace).

5. Install AppPage Builder in your database. ▶ Install AppPage Builder in Your Database.

6. Setup AppPage Builder on your web server. ▶ Setup AppPage Builder on Your Web Server.

7. Create a sample AppPage. ▶ Create Sample AppPage.

Run the sample application.

8. Enter the URL `http:// your_server/scripts/webdriver.exe`.



This How To has been compiled into two separate files for ease of printing. The basic file contains all of the steps you need to Create Web Applications with AppPage Builder. The secondary file contains additional detailed instructions for setting and testing database environment properties.

FIG. 19B

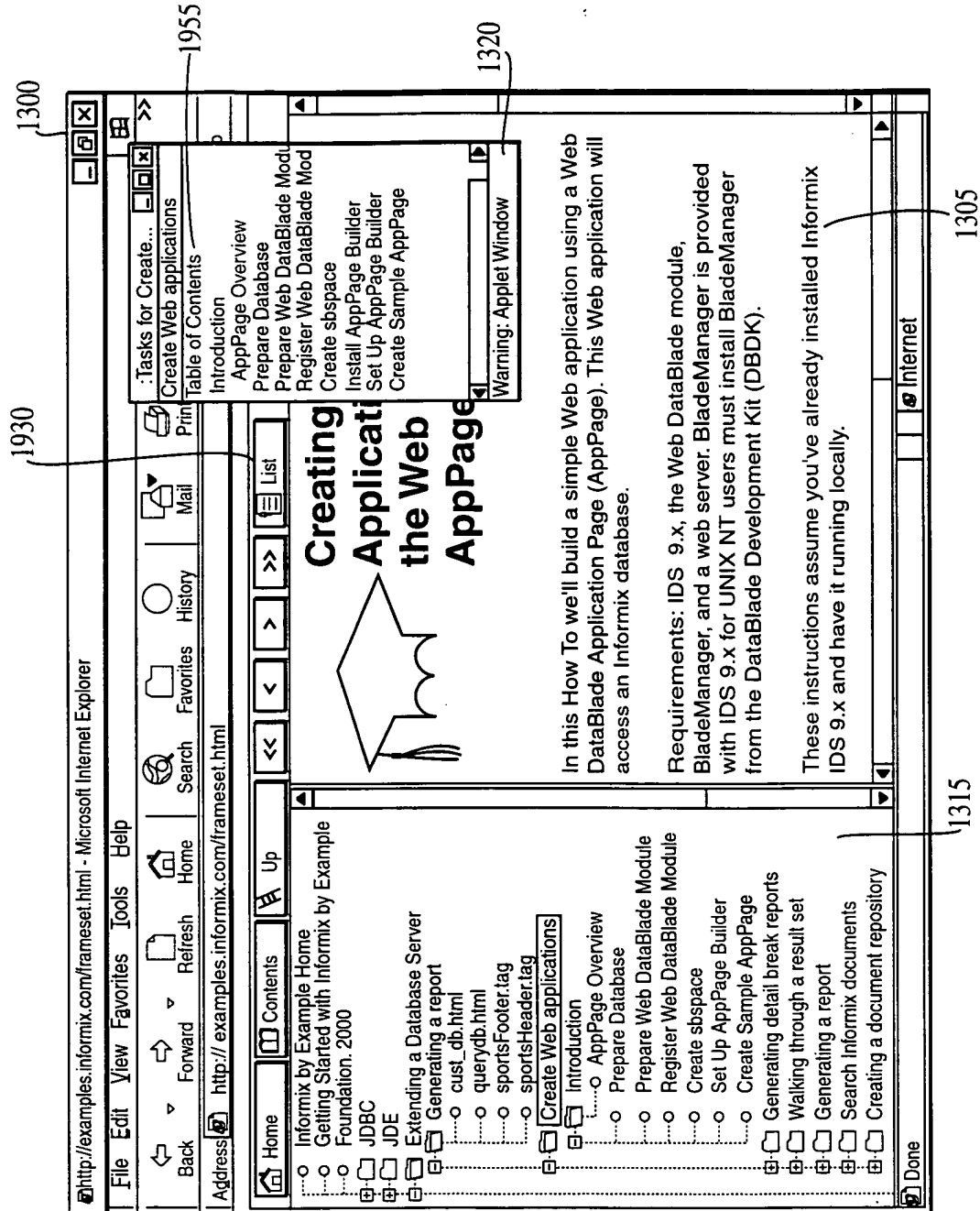


FIG. 19C

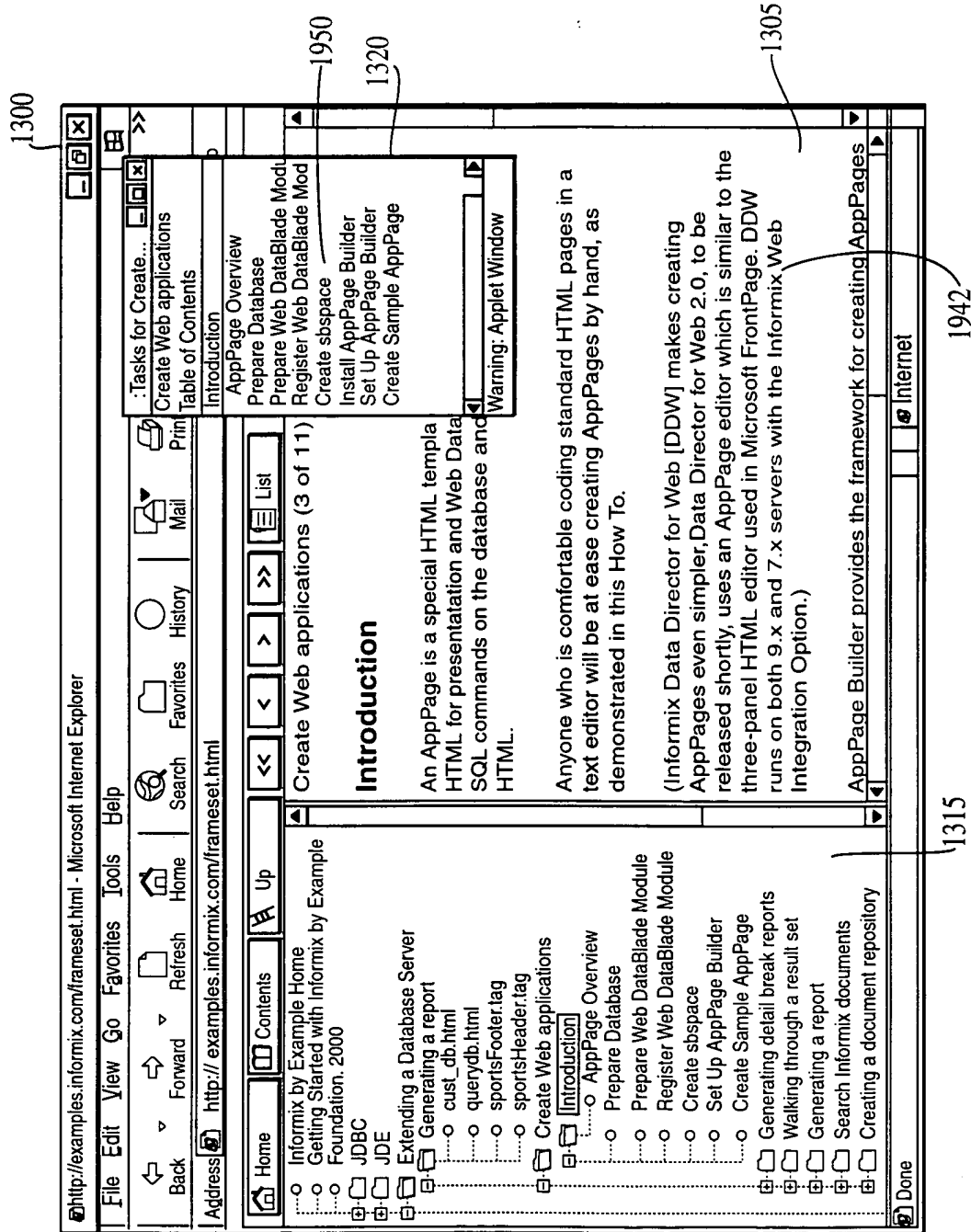


FIG. 19D

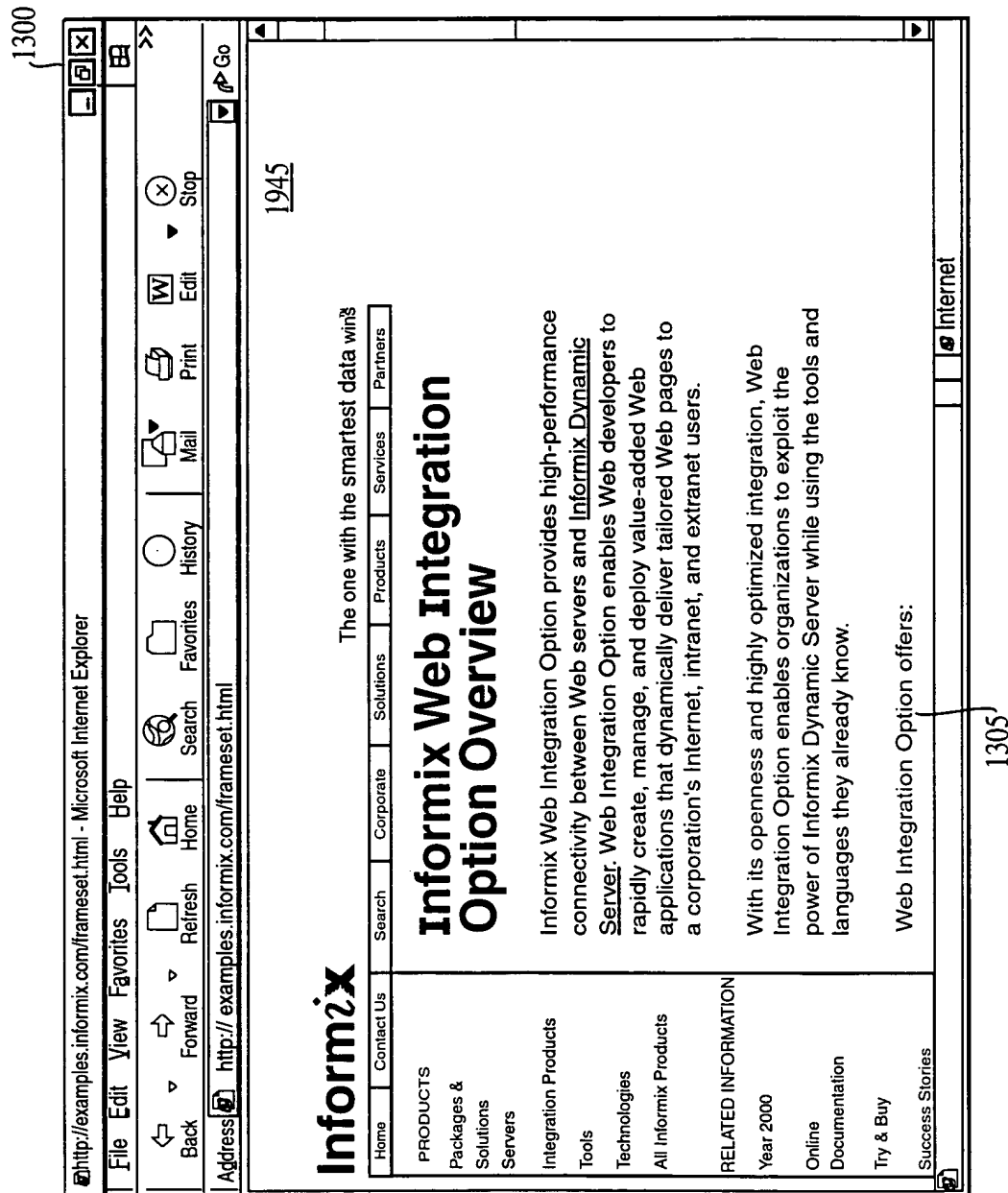


FIG. 19E

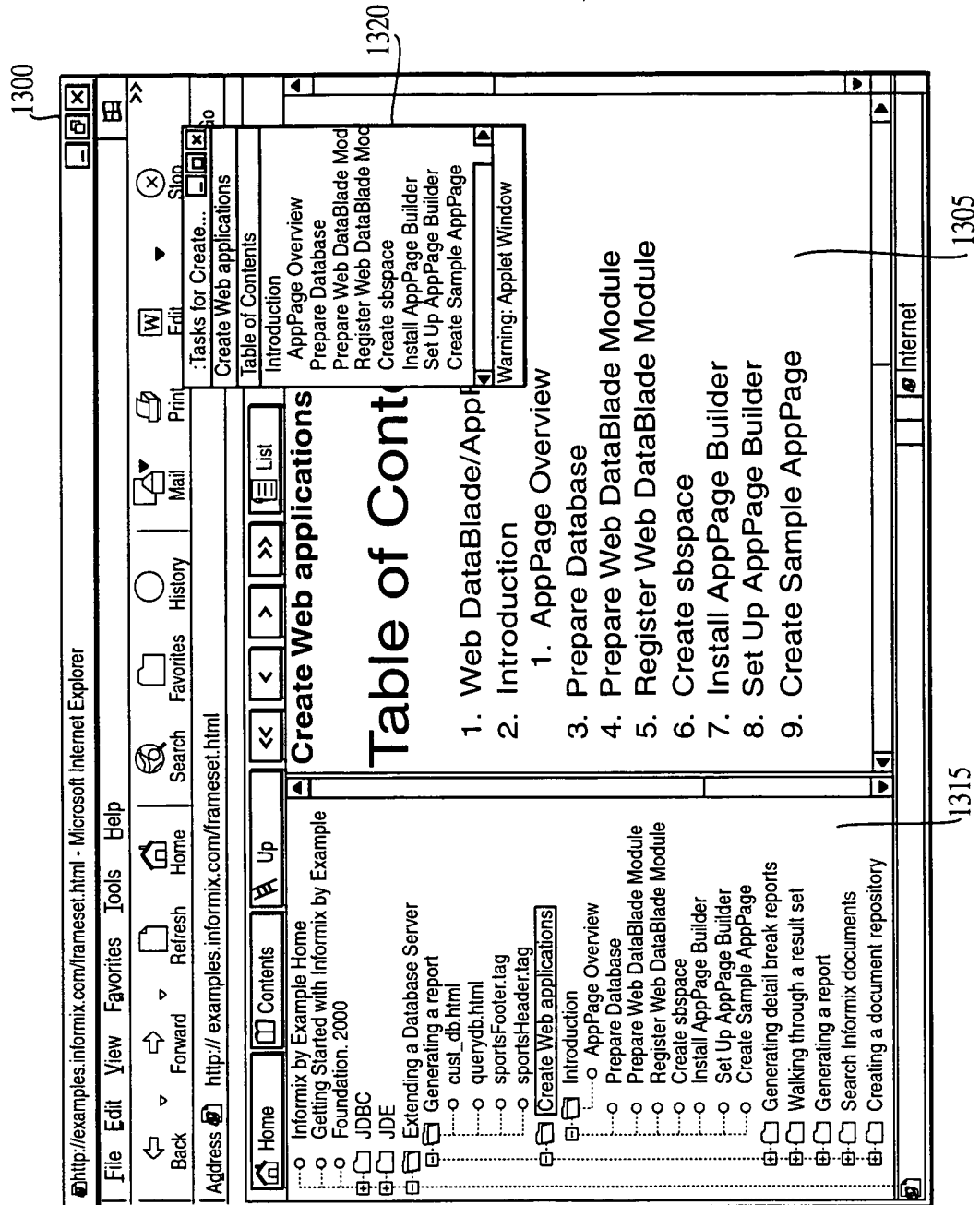


FIG. 19F

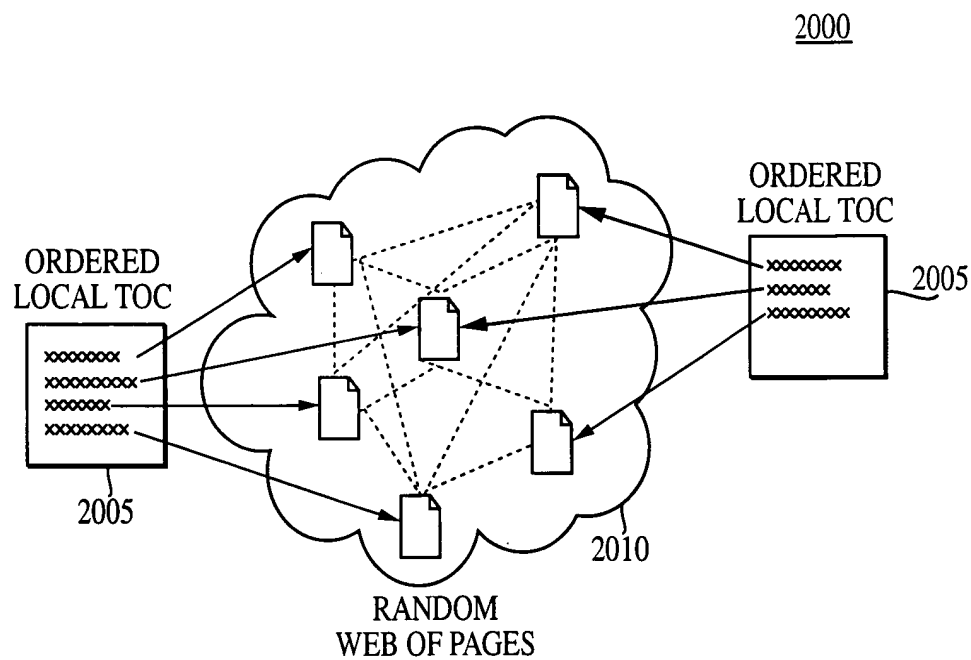


FIG. 20